EINFÜHRUNG IN DIE MIKRO-PROZESSOR-TECHNIK

TEXAS INSTRUMENTS
Deutschland GmbH

Lexikon der Mikroelektronik

Der Informationsbedarf – speziell auf dem Mikroelektronik-Sektor – wird täglich größer. Die Vielfalt an Informationsmaterial, hauptsächlich in englischer Sprache, wird immer umfangreicher. Bei einer sich explosionsartig entwickelnden Technik wie der Mikrocomputer-Technik, die nahezu täglich neue Produkte und neue Anwendungsmöglichkeiten liefert, ist naturgemäß auch die zugehörige Terminologie ständig im Fluß. Während einige Ausdrücke bereits zur Standard-Terminologie gehören, kommen ständig neue Begriffe hinzu.

Da diese neuen Begriffe Autoren und Firmen bisweilen auch unterschiedlich interpretieren, wird der Anwender ständig mit großen Schwierigkeiten konfrontiert.

Das neue LEXIKON DER MIKROELEKTRONIK mit über 5.000 Stichwörtern und deren exakten Erläuterung beseitigt diese Schwierigkeiten und schließt eine große Lücke in dem vorhandenen Literaturangebot.

Dieses neue Standardwerk ist mehr als nur ein Dictionary. Es ist ein zuverlässiger Führer durch die Bereiche Mikroelektronik und -computertechnik und bietet jedem – Elektronikspezialist oder Amateur – einen wirklich tiefen Einblick in die Gebiete der modernen und neuzeitlichen Elektronik.

Das Lexikon ist so aufgebaut, daß es gleichzeitig als umfassendes Englisch-Deutsches Fachwörterbuch dient. Den alphabetisch geordneten englischen Fachausdrücken folgt zunächst die entsprechende deutsche Übersetzung, an die sich eine präzis abgefaßte Erläuterung des Begriffes anschließt.

Bestell-Nr. TM 2138 , DM 137,00*
Einführungspreis bis 31. Oktober '77:
DM 109,00*

*Unverbindliche Preisempfehlung in DM/Stck. inkl. MwSt., zuzügl. Versandspesen.

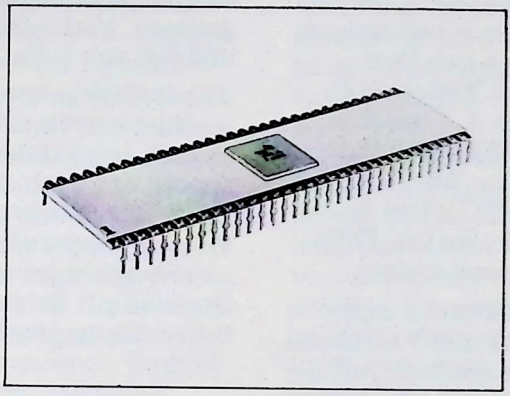
EINFÜHRUNG IN DIE MIKROPROZESSOR-TECHNIK

Grundlagen/Entwurfsprinzipien/Anwendungen

LEARNING CENTER

Haggertystr. 1, 805 Freising · Telefon 08161/80-1
Publikation – CEE + CC

EINFÜHRUNG IN DIE MIKROPROZESSOR-TECHNIK



Eine populäre, leicht verständliche und zum Selbststudium geeignete Einführung in die Mikroprozessor-Technik. Für jeden, der verstehen möchte, wie Mikroprozessoren funktionen, was Hard- und Software ist, wie man den geeigneten Mikroprozessor für seine Anwendung auswählt und welche Vorteile sich aus dem Einsatz in Bereichen, wie Terminals, Steuerungen, Datenübertragungssystemen, Luft- und Raumfahrt, Automobilen und Unterhaltungselektronik ergeben. Zum Verständnis der 16 Kapitel dieses Buches sind allgemeine Kenntnisse der Digitalelektronik ausreichend.

Impressum

Die englisch geschriebene Originalfassung dieses Buches stammt von einem achtstündigen Videoband-Kurs, der im Learning Center von Texas Instruments in Dallas unter der Regie von Don L. Cannon, PhD. entstand. Weitere Mitwirkende waren George A. Consolver, BS; Gerald Luecke, MSEE; Jack P. Mize, PhD; Robert N. Sherman, MSCS; William D. Simpson, MSEE; Dan F. Thomas, AET; Roger S. Walker, PhD; John G. Wester, BSEE und andere.

Dieser Kurs ist ebenfalls in der englischen Fassung, als Buch und Videoband, von Texas Instruments Deutschland GmbH erhältlich.

Texas Instruments Deutschland GmbH (TID) übernimmt keine Gewähr dafür, daß die beschriebenen Schaltungen, Baugruppen, Verfahren usw. funktionsfähig und frei von Schutzrechten Dritter sind.

Alle Rechte an diesem Werk sind TID vorbehalten. Ohne ausdrückliche Genehmigung von TID ist es auch nicht gestattet, das Buch oder Teile daraus in irgendeiner Form durch Fotokopie, Mikrofilm oder ein anderes Verfahren zu vervielfältigen oder zu verbreiten. Dasselbe gilt für das Recht der öffentlichen Wiedergabe.

© Copyright 1977 by
Texas Instruments Deutschland GmbH

Herausgeber:
Texas Instruments Deutschland GmbH,
Learning Center, 805 Freising
Haggertystr. 1, Tel. 0816/80-1

Gesamtherstellung:
Technik Marketing, München
TM 1719/877 1. Auflage
Printed in Germany

ISBN 3-88078-009-9

VORWORT

Unter dem Schlagwort "Mikroprozessor" verbirgt sich eine technische Revolution, die oft mit der Erfindung des Elektromotors verglichen wird, und die die gesamte Struktur vieler mechanischer, elektronischer und elektromechanischer Produkte in den kommenden Jahren drastisch verändern wird. Der Grund dieser Auswirkungen ist in der Entwicklung hochkomplexer elektronischer Halbleiter-Schaltungen (VLSI = very large scale integration) begründet, welche es derzeit gestatten 10.000 Transistoren und mehr auf wenige mm^2 Halbleiterfläche zu integrieren. Deshalb ist es möglich geworden, komplexe Funktionen eines Computers, wie Zentraleinheit, Speicher und Ein-/Ausgabeelemente, in Form Integrierter Schaltungen (IC's) zu realisieren. Das heißt, der Elektronikentwickler erhält Computerleistung auf kleinstem Raum, die als Standardelemente in großen Mengen zu geringen Kosten gefertigt werden können, was bei anwendungsspezifischen VLSI-Schaltungen wegen der geringen Anwenderzahl nur in Ausnahmefällen wirtschaftlich möglich ist. Die Realisierung der Anwendung wird beim Mikroprozessor von der reinen Verdrahtung in das Programm, oder auch Software genannt, verlagert, die in Speicherelementen wie RAM's, ROM's oder EPROM's gespeichert werden. Diese Elemente können wiederum als Standard-Bausteine in großen Mengen kostengünstig gefertigt werden. So sind z.B. derzeit bereits Mikroprozessoren erhältlich, welche eine größere Leistungsfähigkeit besitzen als die ersten, mit Elektronenröhren bestückten Rechenanlagen der 50er Jahre, aber deren Größe um mehr als den Faktor 1000 verringert wurde, bei gleichzeitiger Kostenreduzierung auf ca. 1/5000. des ursprünglichen Preises.

Dieses Buch wurde deshalb für einen weiten Leserkreis geschrieben, der im Selbststudium einen umfassenden Überblick über dieses komplexe Gebiet gewinnen möchte und bisher nur über allgemeine Kenntnisse der Digitaltechnik verfügt. Für den Leser ist dabei das wichtigste Ziel, daß er sich sozusagen "ein Gefühl" für die Einsatzmöglichkeiten des Mikroprozessors erarbeiten kann. Deshalb wird in mehr als 8 Kapiteln detailliert auf den Einsatz und Einsatzkriterien in verschiedenen Anwendungsbereichen eingegangen. Die dabei aufgezeigten Lösungsmöglichkeiten mit verschiedenen Mikroprozessoren dienen dabei dem Verständnis der Unterschiede der vielen auf dem Markt erhältlichen Typen und erhebt nicht den Anspruch der Vollständigkeit im Sinne einer Marktübersicht.

Wir haben unser Ziel mit diesem Buch erreicht, wenn Sie die Antwort der "Spezialisten" auf die Frage: "Was kann ein Mikroprozessor? " verstehen, die da heißt: "Praktisch alles, es ist nur eine Frage der verfügbaren Zeit!".

Die Redation
Freising, Mai 1977

INHALTSVERZEICHNIS

Seite

KAPITEL 1 11
Die Vorgeschichte der Mikroprozessoren führt über die Architektur digitaler Rechnersysteme

KAPITEL 2 27
Chip-Architektur

KAPITEL 3 47
Technologien zur Herstellung von Mikroprozessor-Chips

KAPITEL 4 67
Wie ist ein Mikroprozessor einzusetzen?

KAPITEL 5 85
Software für Mikroprozessoren

KAPITEL 6 99
Befehlsvorrat

KAPITEL 7 117
Interface-Entwurf

KAPITEL 8 139
Potenzielle Anwendungen und Grenzen

KAPITEL 9 157
Mikroprozessoren in Steuerungen

KAPITEL 10 173
Mikroprozessor-Anwendungen: Der Entwurf von Datenübertragungssystemen

KAPITEL 11 195
Anwendungsgebiete für Mikroprozessoren: Verkaufsstellen-Terminals

Seite

KAPITEL 12 215
Mikroprozessoranwendungen andere Terminal-Arten

KAPITEL 13 235
Der Mikroprozessor in Avioniksystemen

KAPITEL 14 251
Mikroprozessoren in Automobilen

KAPITEL 15 271
Mikroprozessoren in der Unterhaltungsindustrie

KAPITEL 16 289
Zukünftige Trends

ANHANG A 311
Architektur und Befehlssätze der TI-Mikroprozessoren

ANHANG B 339
Mikroprozessor-Projektablauf und Auswahlliste

ANHANG C 343
Deutsche Literaturhinweise
Englische Literaturhinweise

ANHANG D 351
Wichtige Begriffe (Deutsch/Englisch)

AUFBAU DIESES BUCHES

PHASE I

GRUNDLAGEN

- System-Architektur
- Chip-Architektur
- Herstellungsverfahren

PHASE II

ENTWURFSPRINZIPIEN

- Wie ist ein Mikroprozessor einzusetzen?
- Software und Hardware
- Befehlssätze
- Schnittstellen (Interface)
- Potentielle Anwendungen und Grenzen

PHASE III

ANWENDUNGEN

- Computer und Terminals
- Datenübertragungssysteme
- Luft- und Raumfahrt
- Automobilelektronik
- Unterhaltungselektronik

PHASE IV

ZUSATZINFORMATIONEN

- Trends
- Architektur und Befehlssatz der TI-Mikros
- Mikroprozessor-Projektablauf und Auswahlliste
- Literaturhinweise (Deutsch und Englisch)
- Wichtige Begriffe (Deutsch/Englisch)

Der Aufbau des Buches kann in vier Phasen eingeteilt werden, wie die linke Schematik zeigt.

Phase I soll dem Leser die erforderlichen Grundlagen über die Entwicklung der Systemarchitekturen vermitteln, die Basis der Chip-Architekturen sind wie sie heute bei Mikroprozessoren anzutreffen sind, sowie Herstellungsverfahren der Halbleiterbausteine erläutern.

Die Phase II hat das Ziel dem Leser das Wissen zu vermitteln, welches notwendig ist Mikroprozessorsystemanforderungen zu definieren, Einsatzmöglichkeiten und Grenzen des Mikroprozessors zu erkennen und Hard- und Softwareentwicklungen zu ermöglichen.

Die Phase III dient im wesentlichen dazu beim Leser ein richtiges "Gefühl" für den möglichen Mikroprozessoreinsatz zu entwickeln und das in Phase II vermittelte Wissen praktisch an Hand verschiedener wesentlicher Anwendungsbereiche, wie Terminal, Datenübertragungssysteme ect. zu vertiefen.

Die letzte Phase, Phase IV, liefert dem Leser eine Reihe umfangreicher Zusatzinformationen, wie zukünftige Entwicklungen (Trends), detaillierte Beispiele für Befehlssätze von Mikroprozessoren (z.B. TMS 9900), Mikrocomputern (z.B. TMS 1000) oder Mikroprozessorelementen (z.B. SBP 0400), oder den Mikroprozessor-Projektablauf und Auswahlliste.

Die umfangreichen Literaturhinweise auf deutsche und englische Unterlagenquellen sind zur weiteren Vertiefung des interessierten Lesers gedacht und eine abschließende Sammlung wichtiger Begriffe (Deutsch/Englisch) dient zur Wiederholung bzw. als Nachschlagewerk.

KAPITEL 1

DIE VORGESCHICHTE DER MIKROPROZESSOREN FÜHRT ÜBER DIE ARCHITEKTUR DIGITALER RECHNER-SYSTEME

Dieses Kapitel erläutert die zum Verständnis des gesamten Buches notwendigen Grundlagen. Darüber hinaus wird die Entwicklung des Mikroprozessors, ausgehend von den ersten Rechenanlagen auf der Basis elektronischer Röhren, bis zu seiner gegenwärtigen Form, verfolgt.

Die Vorgeschichte der Mikroprozessoren führt über die Architektur digitaler Rechner-Systeme

In diesem Kapitel wird erklärt, warum der Mikroprozessor – so wie wir ihn heute kennen – entstanden ist. Außerdem wird sein Entwicklungsweg aufgezeigt, der bei den ersten, mit Röhren bestückten Rechenmaschinen, beginnt.

Die Überschrift dieses Kapitels enthält das Wort "Mikroprozessor". Für manche Leute ist ein Mikroprozessor und ein Mikrocomputer das gleiche. Für andere dagegen haben die Wörter Mikroprozessor und Mikrocomputer eine unterschiedliche Bedeutung. Für sie ist "Mikroprozessor" ein weitgefaßter Gattungsbegriff, der ein besonders kleines elektronisches System beschreibt, das geeignet ist, spezifische Aufgaben wahrzunehmen. In diesem Sinne ist ein Mikrocomputer ein Gerät, das durch die Verwendung eines Mikroprozessors entstanden ist.

Die Großintegration
(Bild 1.1)

Die heutigen Großintegrationtechniken (large scale integration) gestatten Integrierte Schaltungen mit mehr als 10.000 Transistoren. Dadurch ergibt sich die Möglichkeit, die hochkomplexen Funktionen eines Prozessors oder Computers auf wenigen mm^2 Halbleiterkristallfläche zu integrieren. Der Begriff "Mikro"-Prozessor oder "Mikro"-Computer ist dadurch entstanden. Bild 1.1a zeigt ein Halbleiterkristall in 20facher Vergrößerung, welches einen Mikroprozessor darstellt und ca. 11 000 Transistoren enthält. Diese winzigen Kristalle (Chips) werden zur einfachen Verarbeitung in Gehäusen mit 18 bis 64 Anschlüssen untergebracht. Bild 1.1b zeigt einen Mikroprozessor in etwa Originalgröße mit 40 Anschlüssen, welches eine Fläche von nur 8 cm^2 einnimmt.

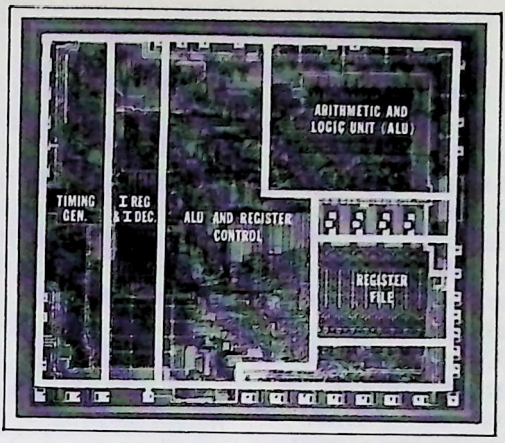


Bild 1.1a
Halbleiterkristall eines Mikroprozessors

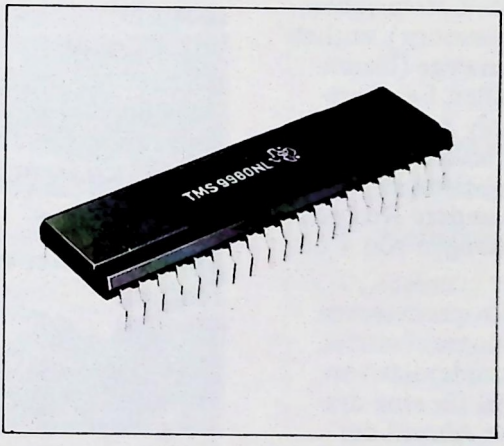


Bild 1.1b
Integrierte Schaltung mit Mikroprozessor

Merkmale eines Mikroprozessors
(Bild 1.2)

Eine treffendere Art, den Begriff Mikroprozessor zu umreißen, ist, seine Merkmale zu beschreiben (Bild 1.2). Das Herz eines Mikroprozessors ist eine integrierte Schaltung. Ein Mikroprozessor kann aus nur einem einzigen integrierten Baustein bestehen oder er kann bis zu 30 oder mehr Bausteine umfassen (Ein- bzw. Mehrchip-Prozessoren). Funktionell enthält ein Mikroprozessor ein programmierbares Datenverarbeitungssystem, das zumindest aus einem Rechenwerk, auch ALU (arithmetic logic unit) genannt, einigen Registern und einer Steuerung besteht. Handelt es sich um einen etwas höher entwickelten Mikroprozessor, dann kann er auch einen Stapelspeicher (Stack), in Form von Registern oder eines Schreib-/Lesespeichers (RAM, random-access memory), enthalten, ein oder zwei Datenwege (Datenbusse) und einen speziellen Festwertspeicher (ROM, read-only memory) für Mikroprogramme oder Mikrobefehle. Mikroprozessoren verarbeiten allgemein kürzere Wortlängen als andere Rechner; gebräuchlich sind Wortlängen von 4 bis 16 Bit.

Dies bedeutet, daß Mikroprozessoren für jede Art des Schaltungsentwurfes, sofern eine gewisse Komplexität vorhanden ist, das Potential für eine drastische Reduzierung der Anzahl der Bauelemente und für eine gleichfalls drastische Verkürzung des für den Schaltungsentwurf benötigten Zeitraums besitzen. Tatsächlich sehen viele Leute in den Mikroprozessoren die Repräsentanten der nächsten Generation digitaler Bausteinsysteme und meinen, daß die Mikroprozessoren die beste Näherungslösung eines neuen Typs von integrierten digitalen Schaltungen darstellen, der auf einer Systemebene betrachtet werden kann. Aus diesen Gründen erregen Mikroprozessoren eine derartige Aufmerksamkeit.

Der Ursprung des Mikroprozessors
(Bild 1.3)

In Bild 1.3 ist der Weg abgebildet, der uns zu dem heutigen Stand auf dem Gebiet der Mikroprozessoren gebracht hat. Er beginnt im Jahre 1948, in dem der Transistor erfunden wurde. Ohne diese Erfindung hätte die Evolution zu den heute bekannten digitalen Systemen nicht stattfinden können.

Kurz nach der Erfindung des Transistors wurde die erste der programmgesteuerten Rechenmaschinen entwickelt. Eine davon war die in Bild 1.3 erwähnte EDVAC. Sie war eine von der University of Pennsylvania im Jahre 1951 entwickelte Rechenmaschine mit Elektronenröhren.

Eine typische EDVAC-Anlage
(Bild 1.4)

Wie durch Bild 1.4 deutlich wird, hatten die EDVAC-Anlagen riesige Ausmaße. Gewöhnlich nahm allein der Prozessor dieser Anlagen, d.h. das Rechenwerk und dessen Steuerung, einen ganzen Raum ein.

Merkmale der ersten Röhrenrechner
(Bild 1.5)

Die EDVAC-Anlagen hatten einige eindrucksvolle Eigenschaften, wie in Bild 1.5 gezeigt wird. Diese Rechenmaschinen wurden gewöhnlich für spezielle Anwendungen in der Wissenschaft eingesetzt.

- Besteht aus 1 bis zu 30 integrierten Schaltungen
- Enthält:

Rechenwerk (ALU)
Register
Leitwerk (Steuerung)
Stapelspeicher (Stack) aufgebaut mit Registern oder einem Schreib/Lesespeicher (RAM)
Datenwege (Datenbusse)
Festwertspeicher (ROM) mit Mikroprogrammen oder Mikrobefehlen

- Kürzere Wortlängen: 4 bis 16 Bit

Bild 1.2
Mikroprozessor Charakteristik

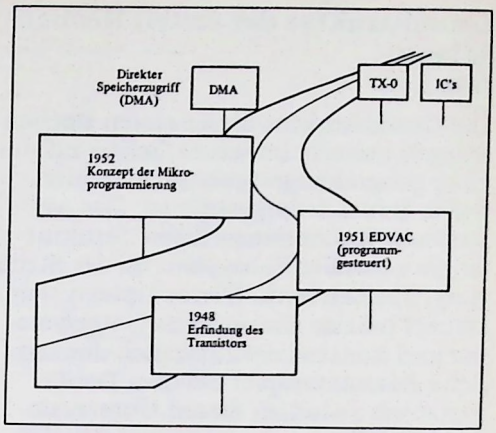


Bild 1.3
Der Ursprung des Mikroprozessors

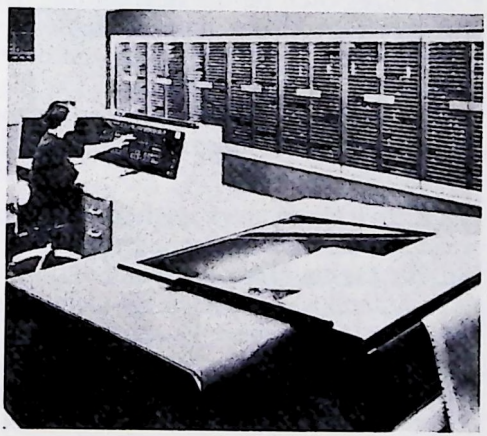


Bild 1.4
Eine typische EDVAC-Anlage

Befehlszykluszeit: 50 μs

Taktfrequenz: 0,5 MHz

Wortlänge: 16 Bit

Speicher: 1 K Worte

Bild 1.5
Merkmale der ersten Röhrenrechner

Grundstruktur der ersten Rechenanlagen
(Bild 1.6)

Die Grundstruktur dieser ersten Rechenanlagen basierte im wesentlichen auf der einer programmgesteuerten Maschine, wie in Bild 1.6 dargestellt ist. Das wesentliche Kennzeichen dieser Struktur ist ein zentrales Subsystem für die Steuerung, das Leitwerk. Dieses Untersystem steuert präzise alle anderen Untersysteme und kontrolliert zusätzlich das zeitliche Zusammenspiel bei dem Datenaustausch zwischen diesen Untersystemen. Das Rechenwerk war gewöhnlich eine einfache arithmetische Einheit, welches nur arithmetische Operationen durchführte. Folglich hatte das zentrale Leitwerk in manchen Fällen auch die Kontrolle über Speicher-, Ein/Ausgabe- und arithmetische Operationen.

Die hohen Kosten für die Register der ersten Rechenmaschine machten es erforderlich, daß jedes Register für mehrere verschiedene Aufgaben benutzt werden mußte. Es war nicht vertretbar, für jedes einzelne Untersystem einen eigenen Satz von Registern vorzusehen. Deshalb war die Rechnerstruktur in einem gewissen Maße durch die Hardwarekosten bestimmt. Außerdem verhinderten die Hardwarekosten, daß zu dieser Zeit solche Methoden wie der direkte Speicherzugriff (DMA, direct memory access) weiter entwickelt wurden.

Dezentralisiertes Leitwerk
(Bild 1.7)

Trotzdem war ein Trend zur Dezentralisierung der Steuerung im Aufbau der Maschinen der fünfziger Jahre festzustellen. Die Dezentralisierung gestattete jedem Untersystem, seine eigene Steuerung zu haben, wobei nur die gesamte Synchronisierung von einer zentralen Steuereinheit stammte. Ein System mit dezentraler Steuerung ist in Bild 1.7 dargestellt. In diesem System wurde die arithmetische Einheit zu einer Arithmetisch/Logischen Einheit (ALU, arithmetic and logic unit) bzw. Rechenwerk mit einer eigenen Steuerung erweitert. Diese ALU mit Steuerung wurde schließlich als Zentraleinheit oder Prozessor der Rechenmaschine bekannt. Der Speicher bekam auf gleiche Art seine eigene Steuerung. Trotzdem mußte die Zentraleinheit immer noch als Bindeglied zwischen dem Hauptspeicher und den Ein/Ausgabekanälen dienen.

Mit dem Erscheinen solcher autonomer Untersysteme und mit der Verwendung des Zentraleinheits-Konzeptes tauchten andere neue Entwicklungen bei den Rechenanlagen auf, wie zum Beispiel Unterbrechungs-Routinen (Interrupt-Routines) zur Erleichterung der Ein/Ausgabe (E/A). Außerdem wurden spezielle E/A-Prozessoren für die Bearbeitung des Austausches von Daten und Befehlen zwischen dem Arbeitsspeicher und den Ein/Ausgabegeräten entwickelt. Für diese Zwecke brauchte dann die Zentraleinheit nicht mehr in Anspruch genommen zu werden.

Die Entwicklung eines solchen Prozessors weist auf das erste Anwendungsgebiet hin, für das ein Mikroprozessor verwendet werden kann, nämlich für kleine spezielle Steuereinheiten. Es sollte festgehalten werden, daß es sich hierbei noch um sehr einfache Steuerungen handelte, die weder arithmetische noch logische Funktionen durchführen konnten, und nur für einfache sequentielle Operationen eingesetzt wurden. Vor der Entwicklung der Unterbrechungs-Verfahren (Interrupt-Verfahren) gab der Prozessor den Ein/Ausgabegeräten ein Signal, wenn er bereit war, Daten anzunehmen oder auszusenden. Durch die Verwendung des Interrupt-Verfahrens wurde es möglich, daß die Ein/Ausgabegeräte dem Prozessor mitteilten, daß sie Daten zu übertragen hatten. Diese Fähigkeit, Unterbrechungen zu bearbeiten, gestattete den Rechenmaschinen, mehr Daten in kürzerer Zeit zu verarbeiten.

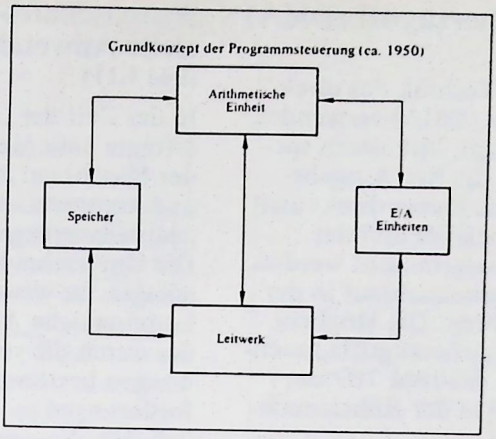


Bild 1.6
Grundstruktur der ersten Rechenanlagen

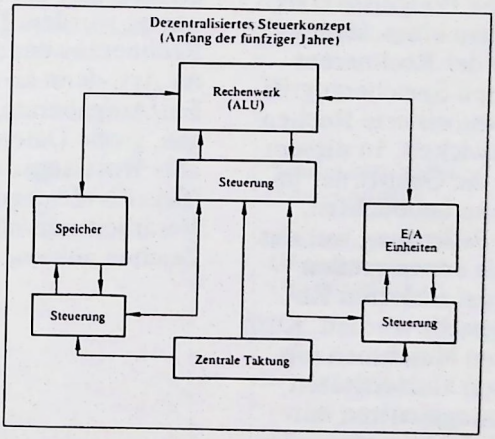


Bild 1.7
Dezentralisiertes Leitwerk

Der direkte Speicherzugriff (DMA)
(Bild 1.8)

Ein System, das die Technik des direkten Speicherzugriffes (DMA) verwendet, wird in Bild 1.8 gezeigt. Mit einem speziellen Prozessor für die Ein/Ausgabe-Steuerung können die Daten direkt zwischen dem Arbeitsspeicher und der Rechnerperipherie ausgetauscht werden, ohne dabei den Programmablauf in der Zentraleinheit zu stören. Die Möglichkeit des direkten Speicherzugriffs,tauchte in Maschinen wie der IBM 709 um 1959 am Ende der Ära der Röhrenrechner auf.

Entwicklung der Rechnerarchitektur von 1950 - 1960
(Bild 1.9)

Etwa während dieser Zeit fanden einige weitere technologische Ereignisse statt. Fast gleichzeitig zu dem einen Meilenstein auf dem Gebiet der Rechnerentwicklung, dem direkten Speicherzugriff, wurde die erste transistorisierte Rechenanlage, die TX-0, entwickelt. In diesem Zeitraum war ferner die Geburt der integrierten Schaltung zu beobachten. Von entscheidender Bedeutung war das Jahr 1960, in dem die ersten großen kommerziellen, transistorisierten Rechenmaschinen hergestellt wurden. Kurz zuvor waren die ersten Maschinen mit neuer Technologie von Universitäten oder anderen Forschungszentren entwickelt worden, und die kommerziellen Versionen erschienen vier oder fünf Jahre später auf dem Markt.

Die Leistungsfähigkeit transistorisierter Rechner (Anfang 1960)
(Bild 1.10)

Die Betrachtung des Leistungsgrades der Maschinen aus diesem Zeitraum zeigt, daß sich fast alle Kenndaten im Vergleich zu den früheren Röhrenrechnern um den Faktor 10 verbesserten.

Wissenschaftliche und kommerzielle Anwendungen
(Bild 1.11)

In der Zeit der frühen sechziger Jahre erfolgte außerdem die Spezialisierung der Maschinen auf wissenschaftliche und kommerzielle Aufgaben. Diese Spezialisierung begann mit der Univac I. Die Unterscheidung zwischen Rechenanlagen für wissenschaftliche und für kommerzielle Aufgaben wurde wegen der durch die verschiedenen Anwendungen bestimmten verschiedenen Anforderungen an die Maschinen erforderlich. Für die wissenschaftlichen Anwendungen benötigten die Maschinen große Wortlängen und schnelle, genau arbeitende Rechenwerke. Andererseits war eine hohe Ein/Ausgabe-Kapazität nicht notwendig, weil die Programme einfach geladen und die Ergebnisse dann abgewartet wurden. Die kommerziellen Rechneranwendungen waren von anderer Art, denn sie erforderten eine hohe Ein/Ausgaberate, relativ kurze Wortlängen, große Datenspeicher und auch variable Wortlängen für verschiedene (Schrift-)Zeichen, wobei es auf gute Verarbeitungsmöglichkeiten für diese Zeichen ankam.

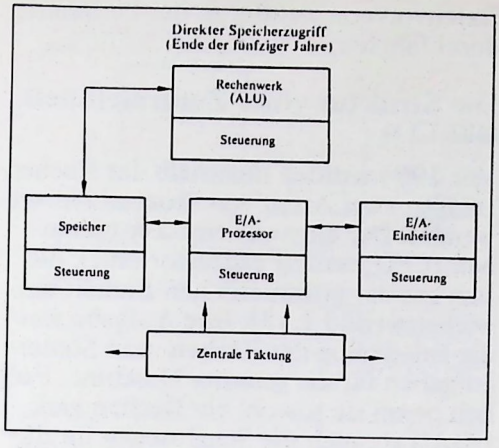


Bild 1.8
Direkter Speicherzugriff

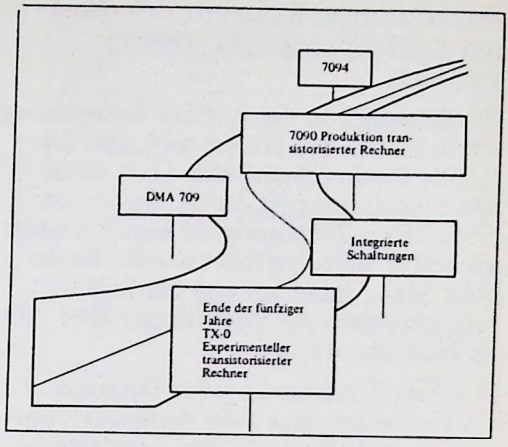


Bild 1.9
Entwicklung der Rechnerarchitektur (Ende der 50er bis Anfang der 60er Jahre)

Befehlszykluszeit: 10 μs

Taktfrequenz: 5 MHz

Wortlänge: 48 Bit

Speicher: 16 K Worte

Bild 1.10
Merkmale transistorisierter Rechner (Anfang 1960)

Wissenschaftliche Anwendungen	Kommerzielle Anwendungen
Numerische Genauigkeit und Geschwindigkeit	Hohe E/A-Kapazität und Zeichenverarbeitung

Bild 1.11
Wissenschaftliche und kommerzielle Anwendungen

Das moderne Konzept des direkten Speicherzugriffs (DMA)
(Bild 1.12)

Unabhängig von der Art der Anwendung setzte sich in den frühen sechziger Jahren die Rechnerarchitektur mit direktem Speicherzugriff durch, wobei am Ende dieses Zeitraums ein sehr hochentwickelter Stand erreicht wurde. Dieser hohe Stand kann anhand der in Bild 1.12 gezeigten Architektur der IBM 7094 verdeutlicht werden.

In dieser Struktur ist jeder Datenkanal ein Prozessor, dem zwei Aufgaben zugeordnet sind: Daten in den Arbeitsspeicher zu schreiben und Daten aus dem Arbeitsspeicher zu einem Ausgabegerät oder einem Sekundärspeicher zu transportieren.

Ein im Speicher vorhandenes Programm für die Bearbeitung des Datentransports steuert den Ein/Ausgabe-Prozessor und wird von ihm aufgerufen. Der Datentransport wird von der Zentraleinheit nur veranlaßt und die Einzelheiten der Übertragung an den Ein/Ausgabe-Prozessor delegiert.

Dieser E/A-Prozessor ist nun das zweite Anwendungsgebiet innerhalb von Rechenanlagen, in dem Mikroprozessoren eingesetzt werden können. Es handelt sich nämlich um einen vollständigen Prozessor mit arithmetischen und logischen Funktionen. Ausgestattet mit solchen Funktionen ist er wesentlich leistungsfähiger als eine einfache Steuerung; dieser Prozessor kann solche Aufgaben durchführen wie Paritätsprüfung, Datenverdichtung (Packen) und Code-Umsetzung.

Schließlich entfalteten sich diese kleinen E/A-Prozessoren zu völlig selbständiger Arbeitsweise, wobei sie nicht mehr von Start- oder Stop-Befehlen der Zentraleinheit abhängig waren. Diese Struktur erschloß den Weg zu Rechnernetzwerken mit E/A-Prozessoren, die eine Datenvorverarbeitung an der Peripherie durchführten.

Die Struktur einer Zentraleinheit
(Bild 1.13)

Vor 1964 wurden innerhalb der Rechenanlagen zwei Arten von Prozessoren verwendet. Der eine war eine Zentraleinheit (CPU, central processor unit), die sich aus der arithmetischen Einheit entwickelte (Bild 1.13). Ihre Aufgabe war die Erledigung der Rechen- und Steueraufgaben für die gesamte Maschine. Folglich besaß sie sowohl ein Rechenwerk (ALU) als auch die Schaltungen für die Ablaufsteuerung und die Entschlüsselung der Befehle.

Ein typischer Ein/Ausgabe-Prozessor
(Bild 1.14)

Die andere Art von Prozessor, der Ein/Ausgabe-Prozessor (Bild 1.14), dient der Steuerung des Informationsflusses vom oder zum Arbeitsspeicher. Folglich trägt der Ein/Ausgabe-Prozessor nur eine minimale Verantwortung für die eigentliche Rechentätigkeit. Er besteht hauptsächlich aus der Schaltung für die Ablaufsteuerung und Befehlsentschlüsselung und aus Registern zur vorübergehenden Zwischenspeicherung der Daten. Im allgemeinen enthielt ein Rechnersystem nur einen einzigen Prozessor, die Zentraleinheit, bis die Technologie der integrierten Schaltungen billige Register hervorbrachte. Dadurch wurde der Einsatz beider Prozessortypen möglich.

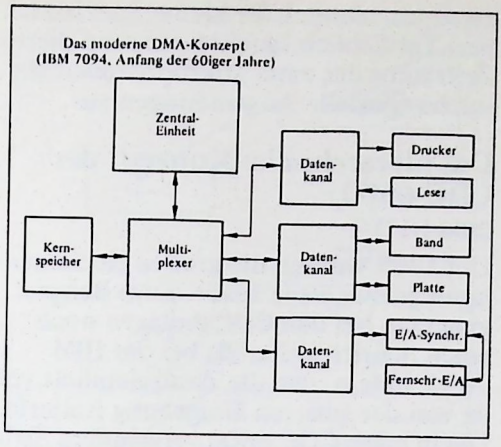


Bild 1.12
Das moderne Konzept für direkten Speicherzugriff (DMA)

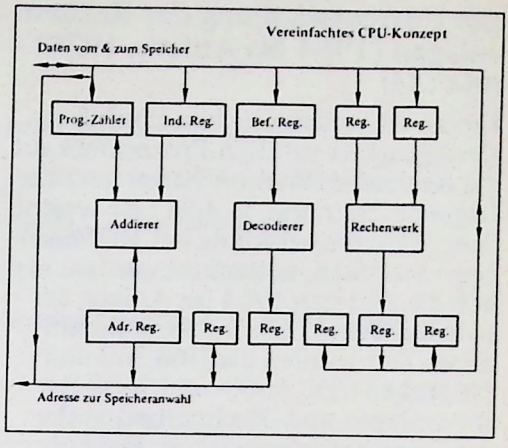


Bild 1.13
Struktur einer Zentraleinheit

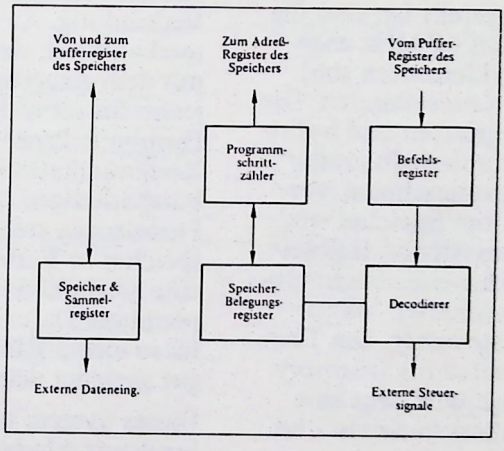


Bild 1.14
Ein typischer Ein/Ausgabe-Prozessor

Die Fortentwicklung der Rechenanlagen (1964 bis Anfang 1970)
(Bild 1.15)

Um zu erkennen, welchen Einfluß die Verfügbarkeit von E/A-Prozessoren auf die Rechnerarchitektur hatte, muß der folgende Zeitraum, in dem eine wesentliche Weiterentwicklung der Rechenanlagen stattfand, betrachtet werden, nämlich die Zeit von 1964 bis Anfang der siebziger Jahre (Bild 1.15). Während dieser Zeit wurden die IBM 360 und die großen CDC 6600- und 7600-Rechenanlagen und -Rechnernetzwerke zur Einsatzreife entwickelt. Die Leistungsfähigkeit dieser Rechnergeneration war wahrhaft eindrucksvoll.

Leistungsfähigkeit der IC-Rechner (Anfang der siebziger Jahre)
(Bild 1.16)

Wie in Bild 1.16 dargestellt ist, sind die Taktfrequenzen bis auf 60 MHz angestiegen; die Befehlszyklenzeiten sind im Bereich von 100 Nanosekunden. Die Technologie der integrierten und hybriden Schaltungen hat innerhalb großer kommerzieller Rechenmaschinen Verwendung gefunden. Ihre Speicher sind auf der Grundlage sowohl von Halbleiter- als auch von Kernspeichertechnologien aufgebaut und enthalten bis zu 262 K Worte Arbeitsspeicher. Die Technik der Speicherunterteilung (memory mapping) wurde eingeführt, was eine Einteilung des Speichers in Seiten und Segmente erlaubte.

Neben der Entwicklung solcher Mammut-Maschinen entstanden in dem Zeitraum von etwa 1965 bis Anfang der siebziger Jahre die Prozeßrechner und andere Spezialrechner. Diese Rechner waren einfachere Versionen der bereits beschriebenen Maschinen. Sie wurden einfach "abgemagert", um spezielle Anwendungslücken kostengünstig schließen zu können. Natürlich stößt man hier auf ein anderes Einsatzgebiet für Mikroprozessoren, nämlich für kleine Spezialrechner. Tatsächlich tauchte während dieses Zeitraums der erste Mikroprozessor für solche spezielle Anwendungen auf.

Das hierarchische Konzept der CDC 6600
(Bild 1.17)

Um 1965 wurden die großen Rechenanlagen größer als je zuvor. Zum Beispiel ging man bei den CDC-Anlagen noch einen Schritt weiter als bei der IBM 7094, indem man die Zentraleinheit völlig von der äußeren Umgebung isolierte. Die in einer CDC 6600 verwendete Struktur ist in Bild 1.17 dargestellt. Der Zentraleinheit sind innerhalb dieser Struktur 10 Untersysteme zugeordnet, die von 24 Arbeitsregistern aktiviert und gesteuert werden. Diese Register stehen ferner mit dem zentralen Arbeitsspeicher mit der Kapazität von 262 K Worten in Verbindung. An dieser Struktur ist bemerkenswert, daß die Zentraleinheit nur mit dem Arbeitsspeicher in Verbindung steht. Sie ist völlig unabhängig von den Peripherie-Prozessoren, die die ganze Kommunikation mit der äußeren Umgebung erledigen. Jeder der Peripherie-Prozessoren steht mit dem großen Hauptspeicher in Verbindung. Wieder liegt hier eine genau definierte Anwendung für die periphere Datenvorverarbeitung vor, wofür in vielen Fällen ein Mikroprozessor gut geeignet wäre.

Dieses System paßt sich wegen seiner Struktur, d.h. hauptsächlich wegen der Unabhängigkeit der peripheren Prozessoren, besonders leicht in ein Rechnernetzwerk ein.

Sofern die externen Einheiten einfache intelligente Terminals mit beschränkten arithmetischen Fähigkeiten sind, haben sie direkten Zugriff auf den Arbeitsspeicher der CDC 6600. Auf diese Weise haben sie schließlich Zugriff auf die Rechenkapazität der Zentraleinheit der CDC 6600.

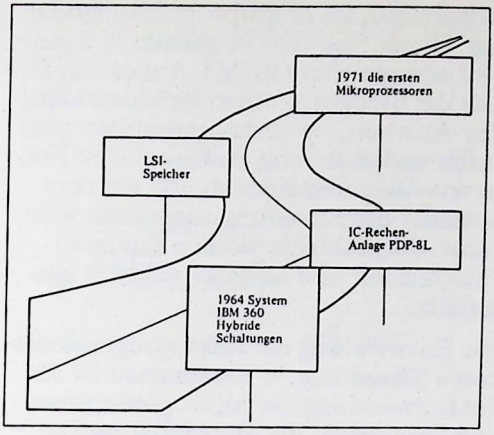


Bild 1.15
Fortentwicklung der Rechenanlagen (1964 bis Anfang 1970)

Befehlszykluszeit: Kleiner 1 μs

Taktfrequenz: Größer als 60 MHz

Wortlänge: Bis 60 Bit

Speicher: Bis 262 K Worte

Speicheraufteilung: In Seiten & Segmente

Bild 1.16
Merkmale der IC-Rechner (Anfang 1970)

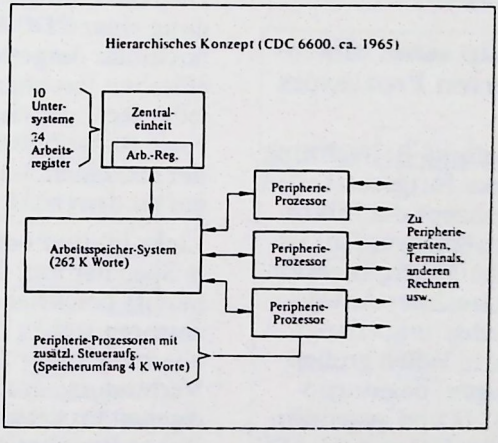


Bild 1.17
Das hierarchische Konzept der CDC 6600

Einsatz der CDC 6600 in einem hierarchischen Netzwerk
(Bild 1.18)

Bild 1.18 zeigt eine Konfiguration, in der eine CDC 6600 das Herz eines Rechnernetzwerkes bildet, dessen äußere Einheiten aus einzelnen Rechnern bestehen. Das spezielle, im Bild dargestellte Netzwerk ist das Netzwerk der University of Texas in Austin. Es besteht aus einer zentralen CDC 6600, an die andere Rechner angeschlossen sind. Auf diese Weise kann ein einziger zentraler Rechner spezielle Rechenanlagen bedienen, die sich an ganz verschiedenen Standorten befinden. Jede dieser Anlagen kann für eine völlig andere Anwendung eingesetzt werden, und jeder dieser weit entfernten Rechner kann auf den großen Speicher oder die enorme Rechenkapazität der CDC 6600 zugreifen.

Organisationsprinzip eines Mikroprogramm-gesteuerten Prozessors
(Bild 1.19)

Bevor diese chronologische Betrachtung über Rechnernetzwerke fortgesetzt wird, sollte das wichtige Konzept der Mikroprogrammierung diskutiert werden. Es hat auf dem Gebiet der Mikroprozessor-Steuerungen und Minirechner-Anwendungen eine große Bedeutung erlangt und wurde außerdem in vielen großen Allzweck-Rechenanlagen, beginnend 1953 mit der EDVAC II und gegenwärtig in Anlagen wie IBM 360 und TI ASC, angewandt.

Das Organisationsprinzip eines Mikroprogramm-gesteuerten Prozessors ist in Bild 1.19 dargestellt. Bemerkenswert ist, daß der Befehl wie gewöhnlich aus dem Arbeitsspeicher geholt und dann entschlüsselt wird. Trotzdem wird jeder Befehl nicht unter der Leitung einer fest verdrahteten Steuerung, sondern unter der Leitung eines Mikroprogramms ausgeführt. Für jeden auszuführenden Befehl ist im Speicher dieser Steuerung ein entsprechendes Mikroprogramm enthalten. Dieser Speicher ist gewöhnlich ein Festwertspeicher (ROM). Außerdem enthält die Steuerung einen Befehlszähler. Zur Abarbeitung eines bestimmten auszuführenden Befehls läuft nun eine Folge von Mikrobefehlen ab, die aus dem Speicher der Steuerung ausgelesen wird. Diese Mikrobefehle werden ihrerseits entschlüsselt und als Mikroschritte ausgeführt.

Die Entwicklung der mikroprogrammierbaren Steuerung für Prozessoren ist für die Entwicklung der Mikroprozessoren und der Geräte, die Mikroprozessoren enthalten, von größter Bedeutung.

Konzept der hierarchischen Netzwerke
(Bild 1.20)

Die Netzkonfiguration unter Verwendung einer CDC 6600 ist in Bild 1.20 nochmals dargestellt. In solchen hierarchischen Rechnernetzwerken besitzt jeder Rechner eine bestimmte Stufe einer Rangordnung, die die Zuordnung der einzelnen, untergeordneten Rechner zu dem zentralen Rechner angibt.

Links im Bild befinden sich der zentrale Speicher und die Zentraleinheit. Die bereits beschriebenen Peripherie-Prozessoren stehen direkt mit den Speichern mehrerer Satellitenrechner in Verbindung, von denen jeder seinen eigenen Prozessor (CPU) und seine eigene Peripherie besitzt. Ein solches Netzwerk mit den unterschiedlichsten, überall verstreuten Prozessoren ist praktisch für fast jede Rechneranwendung geeignet.

Konzept der verteilten Netzwerke
(Bild 1.21)

Neben solchen hierarchischen Systemen sind neuerdings sogenannte "verteilte Netzwerke" im Entstehen, in denen es überhaupt keine zentrale Rechenanlage gibt. Bild 1.21 zeigt ein typisches ver-

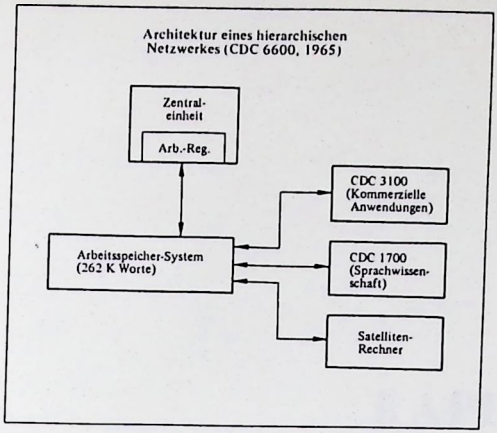


Bild 1.18
Einsatz der CDC 6600 in einem hierarchischen Netzwerk

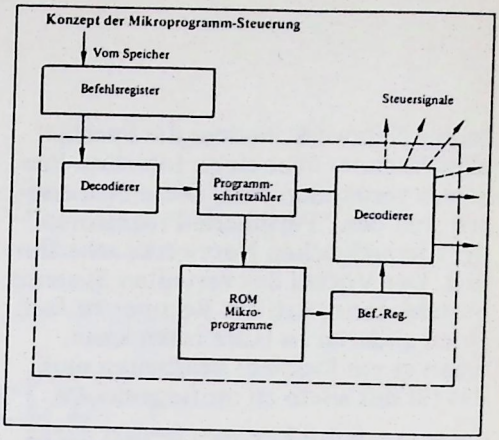


Bild 1.19
Organisationsprinzip eines mikroprogrammgesteuerten Prozessors

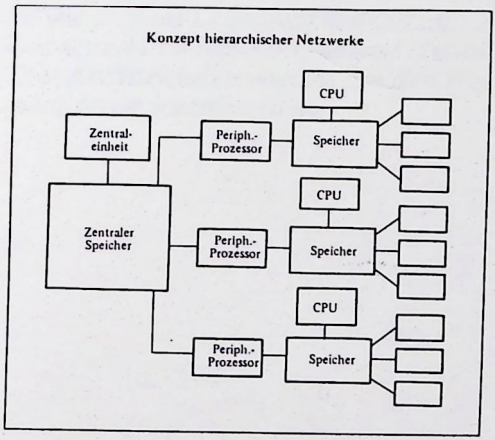


Bild 1.20
Konzept hierarchischer Netzwerke

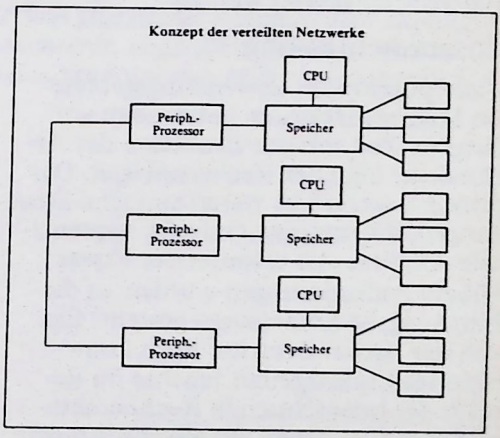


Bild 1.21
Konzept der verteilten Netzwerke

teiltes Netzwerk, in dem die Speicher aller Rechner über einen Interface-Prozessor verbunden sind. Diese Prozessoren sind den "Peripherie-Prozessoren" der hierarchischen Netzwerke sehr ähnlich. Der Vorteil des verteilten Systems besteht darin, daß ein Rechner einfach einen anderen zu Hilfe rufen kann, wenn er ein Problem bearbeiten muß, das für ihn allein zu umfangreich ist.

Dieses verteilte Konzept besitzt bezüglich der Mikroprozessoren eine große Tragweite, weil die weitere Unterteilung der verteilten Netzwerke bis auf die Ebene der Mikroprozessoren weitergeführt werden kann. In einem verteilten System können Mikroprozessoren vielfältig eingesetzt werden.

Zusammenfassung

Viele potentielle Anwendungsgebiete für Mikroprozessoren entstanden vor längerer Zeit mit der Evolution der Architektur digitaler Rechenanlagen. Das erste Einsatzgebiet waren einfache Steuerungen, die nur eine einfache sequenzielle Arbeitsweise erforderten. Etwas höhere Anforderungen wurden an die Ein/Ausgabe-Prozessoren gestellt. Ein von der Anwendung her noch komplexeres Einsatzgebiet tauchte im Bereich der hierarchischen Rechnernetzwerke auf, in denen ein Mikroprozessor ohne weiteres auf der unteren Ebene arbeiten kann.

Ausgehend von diesen Anfängen ist leicht zu sehen, wie der Mikroprozessor seinen Einzug in komplexe Prozeßregelungen und -steuerungen, sowie in numerisch gesteuerte Werkzeugmaschinen und in die Kraftfahrzeugindustrie halten kann, in der Abgas-Vorschriften einen höheren Wirkungsgrad für Motoren nach komplizierteren Regel- und Steuertechniken verlangt.

KAPITEL 2

CHIP-ARCHITEKTUR

Dieses Kapitel untersucht System- und Chip-Architekturen und zeigt deren Einfluß auf die System-Leistungsfähigkeit auf. Zusätzlich gibt dieses Kapitel einen Auszug über grundsätzliche Vorteile und Merkmale mehrerer, augenblicklich erhältlicher Chip-Architekturen wieder, um den Weg für das Studium von Mikroprozessoranwendungen vorzubereiten.

Chip-Architektur

Im vorigen Kapitel wurden die Geschichte der digitalen Systeme und mehrere Tätigkeiten, welche potentielle Mikroprozessoranwendungen darstellen, besprochen. Es ist sicherlich die historisch bedingte Mannigfaltigkeit der digitalen Systeme, daß auf dem heutigen Markt so viele verschiedene Mikroprozessoren zu finden sind und täglich neue hinzukommen.

Dieses Kapitel beschreibt Chip-Architekturen und versucht etwas Ordnung in die Auswahl von Mikroprozessoren zu bringen.

Erinnern wir uns: Es gibt zwei fundamentale Strukturen innerhalb von Digitalrechnern. Eine Struktur hat DMA-Fähigkeit, die andere nicht. Welche der beiden verwendet wird, hängt vom spezifischen verwendeten Baustein-Satz (chip set) ab.

Grundstruktur eines Mikroprozessor-Systems ohne direkten Speicherzugriff
(Bild 2.1)

Bild 2.1 zeigt die heute meistverwendete Struktur, die keinen DMA besitzt. Trotzdem kann in dieser Struktur dezentralisierte Steuerung vorgesehen werden. Die CPU steuert die Kommunikation zwischen Speicher und Ein/Ausgangskanälen. Diese Ein/Ausgangskanäle steuern wiederum die Kommunikation zwischen der Peripherie und der CPU.

Grundstruktur eines Mikroprozessor-Systems mit direktem Speicherzugriff
(Bild 2.2)

Einige Prozessoren besitzen DMA-Fähigkeit, wie sie z.B. Bild 2.2 zeigt. Um diese Struktur zu implementieren, werden normalerweise zwei Chip-Typen verwendet. Ein Chip ist ein zentraler Prozessor, während der andere Chip eine spezielle Speicher-Schnittstelle darstellt, welche die Aufgaben der Ein/Ausgabe bewältigen kann. Für alle diejenigen Anwendungen, die hohe Ein/Ausgabe-Daten-Raten benötigen, wie Kommunikationssysteme oder Mikrorechnernetzwerke, scheint die DMA-Architektur die vielversprechendste zu sein.

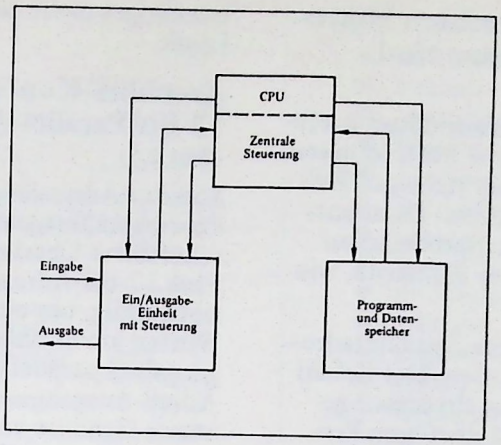


Bild 2.1
Grundstruktur eines Mikroprozessorsystems ohne DMA

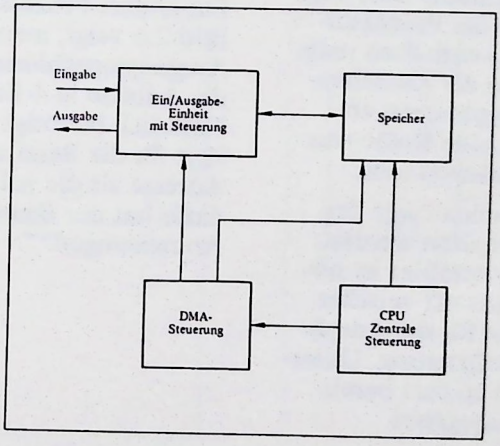


Bild 2.2
Grundstruktur eines Mikroprozessorsystems mit direktem Speicherzugriff (DMA)

Elemente, die den meisten Mikroprozessoren gemeinsam sind
(Bild 2.3)

Unabhängig von der verwendeten Architektur muß der Prozessor Befehle holen, interpretieren und in der richtigen Reihenfolge ausführen können. Dazu enthalten die meisten Mikroprozessoren eine Reihe gemeinsamer Elemente, wie Bild 2.3 zeigt.

Ein Element muß für die Speicheradressierung vorgesehen werden. Der Befehl muß vom Speicher zum Prozessor gesendet werden, er wird dort vom Prozessor im Befehlsregister abgelegt. Außerdem muß eine Befehls-Dekodierlogik vorgesehen werden, die die Bedeutung des Befehls bestimmt und die für die Ausführung des Befehls notwendigen Steuersignale generiert. Manche Befehle verlangen arithmetische oder logische Operation, so daß der Prozessor ein Rechenwerk (ALU) enthalten muß. Andere Befehle können die Zwischenspeicherung von Teilergebnissen einschließen, so daß auch eine Reihe von Arbeitsregistern bereitstehen muß.

Schließlich muß die Reihenfolge der Befehlsadressen festgehalten werden, d.h. ein Programmschrittzähler ist nötig. Weiterhin finden wir oft verschiedene Adreßregister, die für die Adreßmodifikationen wie Indizierung, Unterprogramm-Aufruf und so fort bereitstehen, im Prozessor integriert.

Das Basis-Befehls-Repertoir des Rechenwerks
(Bild 2.4)

In einer kurzen Besprechung der einzelnen Prozessor-Untersysteme werden wir zusätzliche Details kennenlernen. Die Fähigkeiten eines Rechenwerks (ALU) sind in Bild 2.4 aufgetragen. Sie schließen Addition, logisches ODER, logisches UND und arithmetisches Schieben ein. Einige Prozessoren bieten einiges mehr an Arithmetik und Logik.

Anschluß-Konfiguration für einen 12 Bit-Parallel-Ausgang
(Bild 2.5)

Soll der Adressierungsumfang eines Prozessors festgelegt werden, sind wirtschaftliche Überlegungen anzustellen. Eine 12-Bit-Adresse ist zum Beispiel notwendig, um eines von 4096 Speicher-Worten anzuwählen, so daß alle 12 Ausgangsbits parallel als 12 verschiedene Adreß-Ausgangsanschlüsse an dem Prozessor-Gehäuse vorgesehen werden müssen. So eine Konfiguration zeigt Bild 2.5.

Anschluß-Konfiguration für einen 4 Bit-Parallel-Ausgang
(Bild 2.6)

Für andere Prozessoren werden, wie Bild 2.6 zeigt, manchmal nur 4 Adreß-Ausgangsanschlüsse vorgesehen, so daß die Adresse in 4-Bit-Teilen geliefert wird. Natürlich benötigt dies dreimal so viel Zeit für die Bereitstellung einer 12-Bit-Adresse als die voll-parallele Methode; dafür hat der Baustein jedoch kleinere Abmessungen.

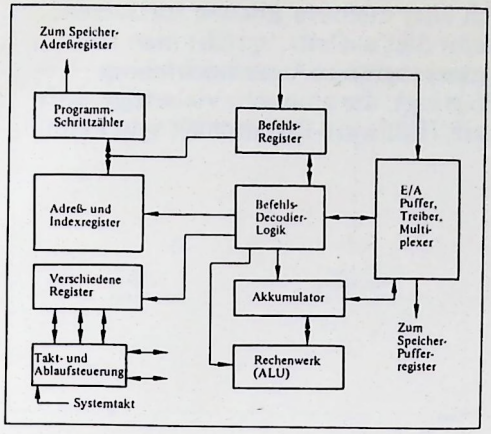


Bild 2.3
Elemente, die den meisten Mikroprozessoren gemeinsam sind

- Subtraktion
- Dezimale Arithmetik
- Multiplikation
- Division
- Exklusives ODER
- Vergleichen
- Schieben
- Serien/Parallel-Umsetzung
- Bit-Manipulation
- Addition
- Logisches UND
- Logisches ODER
- Arithmetisches Schieben

Bild 2.4
Befehls-Repertoire eines Rechenwerks

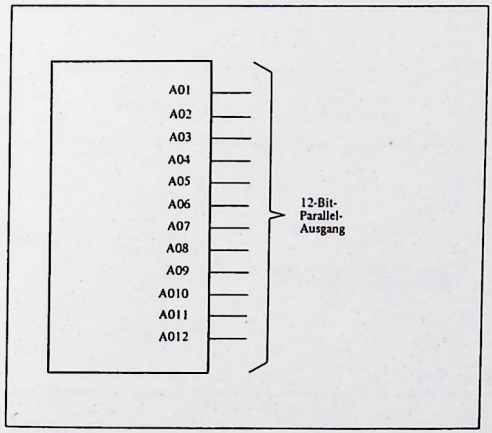


Bild 2.5
Anschluß-Konfiguration für einen 12-Bit Parallel-Ausgang

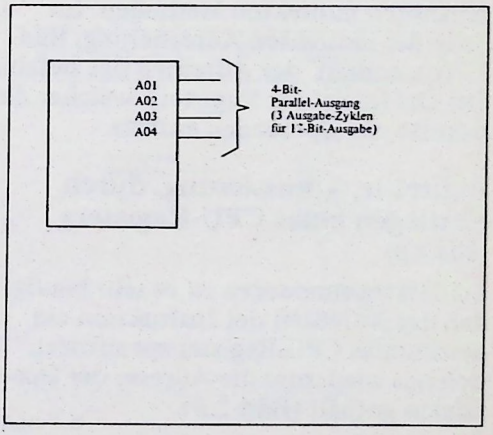


Bild 2.6
Anschluß-Konfiguration eines 4-Bit-Parallel-Ausgangs

Adressier-Techniken
(Bild 2.7)

Es gibt mehrere, in Mikroprozessorsystemen verwendete Adressiertechniken, die kurz erläutert werden sollten. Die direkte Adressierung, Bild 2.7(a), ist die bekannteste Adressierart. Im Falle der direkten Adressierung ist die Adresse des Operanden im Adreßteil des Befehls enthalten, welche dem Prozessor sagt, wo die Daten abgelegt sind. Die relative Adressierung, Bild 2.7(b), ist eine andere Adressierungsmethode. In diesem Fall sind einige der Adreß-Bits Teil des Befehls, der Rest befindet sich in einem der Register des Prozessors. Diese zwei Adressier-Schemata, direkte und relative, kann man "translationale Methoden" nennen.

Neben diesen Arten gibt es eine weitere Gruppe von Adressiertechniken, die sogenannten indirekten Methoden. Im Falle der indirekten Adressierung, Bild 2.7(c), enthält der Adreßteil des Befehls den Ort (meist ein Register), welcher die Adresse des Operanden enthält.

Indirekte Adressierung, durch Festlegen eines CPU-Registers
(Bild 2.8)

Bei Mikroprozessoren ist es sehr häufig, daß der Adreßteil der Instruktion ein bestimmtes CPU-Register spezifiziert, welches wiederum die Adresse der Operanden enthält (Bild 2.8).

Eine andere, bei Mikroprozessoren gängige Form von indirekter Adressierung, ist der Unterprogramm-Aufruf. Er wird verwendet in Unterbrechungs- und komplizierten arithmetischen Abläufen, wie z.B. bei der Multiplikation, um zu dem Speicherplatz zu springen, wo das Mikroprogramm der verlangten Unterroutine gespeichert ist.

Das Abspeichern des augenblicklichen Befehlszähler-Inhalts und das Verzweigen zu einer neuen Unterroutine kann sich über mehrere Ebenen fortsetzen. Wenn dies auftritt, spricht man von Unterprogramm-Verschachtelung (Nesting), die eine sehr vielseitige Software/Hardware-Eigenschaft sein kann.

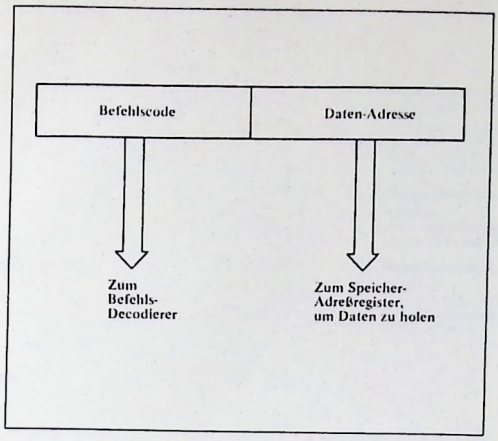
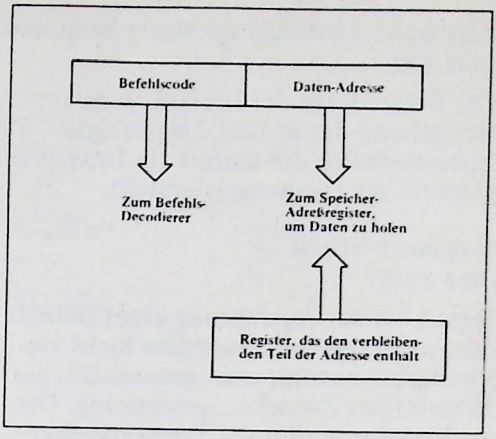
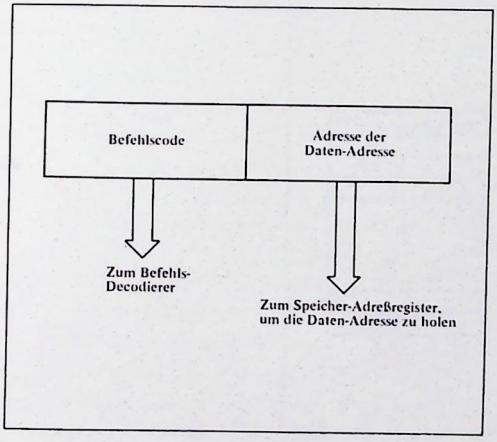


Bild 2.7 (a)
Adressierungstechniken



(b)



(c)

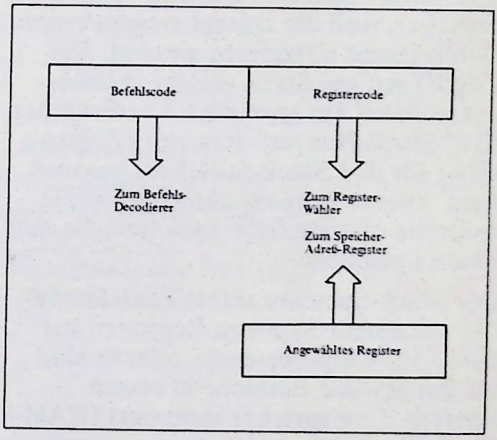


Bild 2.8
Indirekte Adressierung durch Festlegen eines CPU-Registers

Einfache Unterprogramm-Sequenz
(Bild 2.9)

Die Reihenfolge der Ereignisse entspricht weitgehend der in Bild 2.9 gezeigten, die wahrscheinlich die einfachste Unterprogrammfolge (Sequenz) darstellt.

Register-Stack
(Bild 2.10)

Damit bei der Ausführung eines Befehls der Inhalt des Befehlszählers nicht verlorengeht, benützt man gewöhnlich ein Register zur Zwischenspeicherung. Dieses Register wird mit "Stack-Speicher" bezeichnet. Wie der Name bereits andeutet, handelt es sich hier um einen Stapel oder Satz von übereinander liegenden Registern entsprechend Bild 2.10. Eine solche Anordnung als "Stack" nennt man auch "Last-in/First-out"-(LiFo-) Speicher, weil die zuletzt eingegebenen Daten zuerst ausgelesen werden. Der Zugriff auf ein Stack erfolgt üblicherweise durch ein spezielles Adreßregister, das "Stack-Pointer"-Register (Zeigerregister für den Stack-Speicher) genannt wird. Dieses Register kennzeichnet jeweils die oberste freie Speicherzelle des Stack-Speichers.

Der Stack-Speicher selbst kann entweder aus einem Satz von Registern auf dem CPU-Chip bestehen, oder es sind für ihn gewisse Bereiche in einem Schreib-/Lesespeicher reserviert (RAM-Stack).

Wir wollen uns nun mit einigen Mikroprozessor-Architekturen beschäftigen.

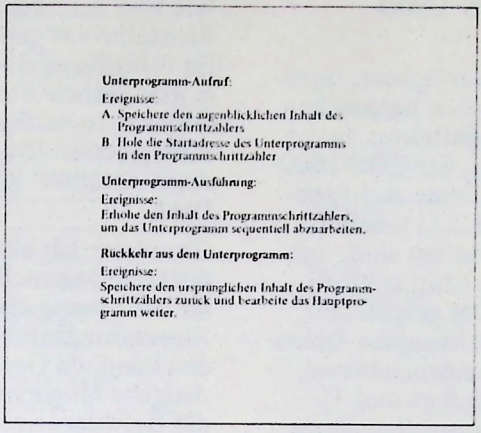


Bild 2.9
Einfache Unterprogramm-Sequenz

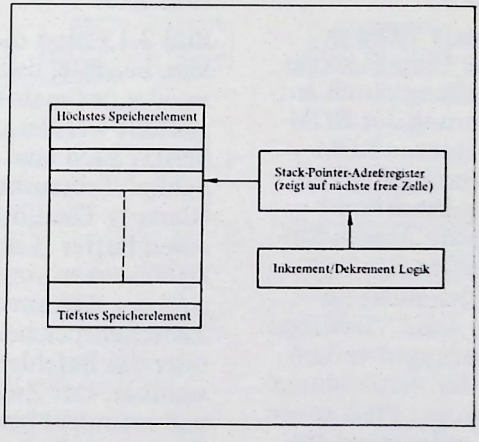


Bild 2.10
Register-Stack

Architektur des TMS 1000
(Bild 2.11)

Nun wollen wir einige verfügbare, repräsentative Mikroprozessoren besprechen und ihre spezifische Architektur analysieren. Einer von diesen, der TMS-1000-Mikrocomputer, besitzt eine recht spezielle Architektur (Bild 2.11). Seine auffallendsten Eigenschaften sind, daß das Programm auf dem Chip in einem 1024 x 8-Bit-Worte-ROM gespeichert ist und daß der Eingabe/Ausgabe-Datenfluß über einen 4-Bit-Eingabebus und über einen 8-Bit-Ausgabebus und 11- oder 13 speichernde Kontrollausgänge erfolgt. Sein 4-Bit-Rechenwerk (ALU) arbeitet parallel. Die Basis-Registerlänge für Akkumulatoren und Datenspeicher beträgt 4 Bit. Der Datenspeicher mit 64 Worten ist ebenfalls auf dem Chip integriert.

Da das Programm in einem ROM gespeichert ist, werden die Chip-Funktionen während der Herstellung durch entsprechende Programmierung der ROM-Masken bestimmt. Ein solcher ROM-Programmspeicher in Verbindung mit diesem Architektur-Typ beschränkt den Einsatz auf spezifische Anwendungen mit großen Stückzahlen.
Dieser Baustein eignet sich nicht für Applikationen, die eine hohe Vielseitigkeit und Programmänderungen verlangen. Er ist jedoch ideal für Anwendungsfälle, bei denen ein günstiger Preis sowie hohe Stückzahlen im Vordergrund stehen und die keine Änderung erfordern.

8008- und 8080-Prozessoren
(Bild 2.12)

Um einen Chip mit mehr allgemeinen Anwendungsmöglichkeiten zu erhalten, müssen externe Programmspeicherung, Interruptmöglichkeiten, DMA-Fähigkeit und parallele Daten-Ein/Ausgabe bereitgestellt werden. Mehrere Prozessoren mit diesen Fähigkeiten sind auf dem Markt, zwei davon sind der 8008- und der 8080-Prozessor.

Wie Bild 2.12 zeigt, haben diese beiden Bausteine viel gemeinsam, wie z.B. die für 8-Bit-Parallelbetrieb ausgelegte ALU, es gibt jedoch auch signifikante Unterschiede. So verfügt beispielsweise der 8080 über 16-Bit-Adressierung, über einen externen RAM-Stack und über DMA.

Grundsätzlich bilden der 8008 und 8080 ein Mikrocomputer-System, das für die Lösung einer großen Vielfalt von Anwendungsproblemen eingesetzt werden kann, da Geschwindigkeit, gute Ein/Ausgabe-Möglichkeit und ein umfassender Befehlsvorrat zur Verfügung stehen.

Um zu sehen, wie diese Fähigkeiten bezüglich der Architektur ausgebildet sind, betrachten wir die Architektur des 8080.

Grundarchitektur des 8080
(Bild 2.13)

Bild 2.13 zeigt die Architektur des 8080. Man beachte, daß im Rechenwerk binäre oder dezimale (BCD) Arithmetik frei gewählt werden kann. Das Rechenwerk besitzt auch eine schnelle Additionsmöglichkeit (vorausberechneter Übertrag (Carry)). Die Ein/Ausgabe wird über einen Puffer (Latch) abgewickelt. Als Datensenken vom Ein/Ausgabebus sind der Akkumulator und eines der Zwischenspeicher-Register für Daten, oder das Befehlsregister für Befehle,anwählbar. Der Zwischenspeicher (scratch pad memory) besteht aus drei 16-Bit-Registern, die in 8-Bit-Subregister organisiert sind. Jedes Subregister kann separat angewählt werden. Diese Architektur enthält außerdem einen 16-Bit-Programmschrittzähler und ein 16-Bit-Stack-Pointer-Register mit zugehöriger Inkrement-/Dekrement-Logik. Die Speicheradressen werden über 16 Anschlüsse zur gleichen Zeit ausgegeben.

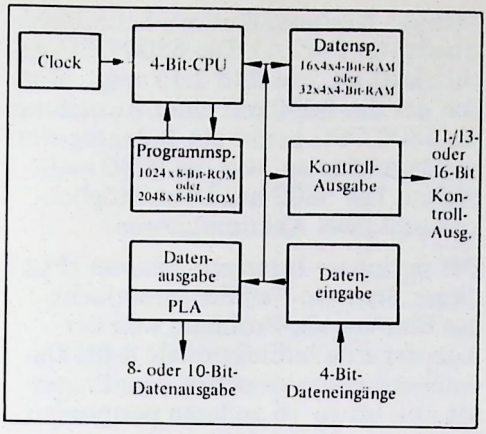


Bild 2.11
TMS 1000 Architektur

	8008	8080
ALU-Wortlänge	8 Bit	8 Bit
Adreßbus	8 Bit	16 Bit
Stack-Speicher	7 interne Register	Externes RAM
BCD-Arithmetik	Nein	Ja
E/A	Interrupt	Interrupt und DMA

Bild 2.12
8008- und 8080-Prozessoren

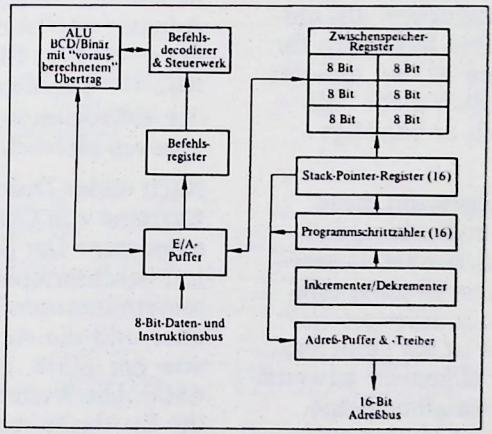


Bild 2.13
Grundarchitektur des 8080

Adressen, Befehle und Daten des 8008

(Bild 2.14)

Um parallele Verfügbarkeit von Daten und Adressen zu erreichen, ist der 8080 als 40poliger Baustein ausgeführt, während der 8008 im 18poligen Gehäuse angeboten wird. Die größte Einschränkung, die sich aus der Beschränkung auf 18 Anschlüsse ergibt, ist, daß die Adressen, Befehle und Daten, wie Bild 2.14 zeigt, alle über ein und denselben, 8 Anschlüsse umfassenden, Ein/Ausgangsbus fließen müssen. Dadurch ist der 8008 viel langsamer als der 8080. Da außerdem der 8008 einen internen Stack-Speicher hat, ist die Unterprogramm-Verschachtelung auf sieben Ebenen beschränkt.

Diese zwei Bausteine sind gute allgemein anwendbare Prozessoren. Ihr einzig möglicher Nachteil in Systemen für spezielle Anwendungen ist der, daß der Anwender wahrscheinlich mehr "Mikroprozessor" kauft als er benötigt oder benutzen will.

Ein anderes Charakteristikum dieser Mikroprozessoren ist außerdem, daß sie Teile einer Bausteinfamilie darstellen, die in einem System in einer relativ einfachen Weise zusammengeschaltet werden kann. Dies ist ein gemeinsames Kennzeichen der allgemein anwendbaren Prozessoren. Spezialbausteine, wie der vorher erwähnte Mikrocomputer TMS 1000, haben normalerweise dieses Merkmal nicht.

Grundarchitektur der 6800 CPU

(Bild 2.15)

Eine Sammlung von Chips, die eine vollständige Bausteinfamilie bilden, sind die Bauelemente der 6800-Mikroprozessor-Familie. Diese Familie besteht aus dem Mikroprozessor, Festwertspeichern (ROM), Schreib-/Lesespeichern (RAM), einem Peripherie-Schnittstellenadapter (PIA) und asynchronen Kommunikations-Schnittstellenadapter (ACIA). Die 6800 CPU-Architektur ist, wie Bild 2.15 zeigt, ähnlich der der 8080 mit einer Ausnahme: die 6800 CPU bietet ein Indexregister auf dem Chip an, was die 8080 nicht besitzt. Die 6800 hat DMA-Möglichkeit und zwei Akkumulatoren.

Der periphere Interface-Adapter (PIA) dieser Baustein-Familie vereinfacht das Ein/Ausgabe-Problem, weil der Adapter eine bidirektionale 8-Bit-Datenübertragung zwischen dem Prozessor und bis zu 16 anderen peripheren Geräten und/oder Mikroprozessoren ermöglicht. Damit bietet dieser Interface-Adapter wesentliche Vorteile für die Verwendung der 6800 in Kommunikationssystemen und Mikroprozessor-Netzwerken. Der separate, für asynchrone Kommunikation geeignete Interface-Adapter (ACIA) ist ebenfalls ein Teil der 6800-Familie. Er löst die Kommunikationsprobleme, die auftreten, wenn der Mikrocomputer an Telefonleitungen über ein MODEM angeschlossen wird.

Nach dieser Diskussion scheinen zwei Extreme von Chip-Architekturen zu existieren: Der Spezialzweck-Prozessor mit beschränktem Einsatzgebiet, mit seinem internen ROM-Programmspeicher und die Allzweck-Prozessoren, wie der 8008, der 8080 und der 6800. Die Weiterentwicklung könnten die Bausteine sein, die Prozessoren mit 16 oder mehr Bits bereitstellen, welche höhere Geschwindigkeit und Genauigkeit erlauben. Als Standard-Datenformat kristallisiert sich 16 Bit heraus; es stellt jedoch keine Grenze nach oben dar.

Eine andere Möglichkeit besteht darin: die 8-Bit-Chips beizubehalten, jedoch deren Vielseitigkeit zu verbessern. Ein Beispiel, das diesen Weg vorzeichnet, ist die Weiterentwicklung des 8008 zum 8080.

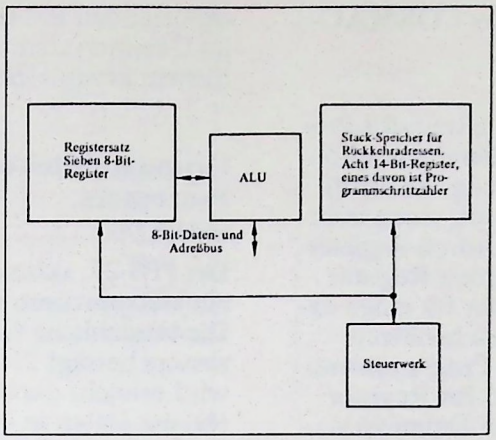


Bild 2.14
Adressen, Befehle und Daten des 8008

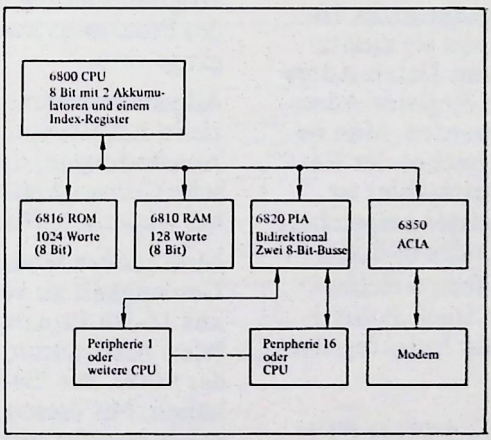


Bild 2.15
Grundarchitektur des 6800

Grundarchitektur des COSMAC-Prozessors
(Bild 2.16)

Der in Bild 2.16 dargestellte 8-Bit-Prozessor verfügt über die bereits besprochene Indizier-Einrichtung. Er bietet außerdem die Möglichkeit, einen internen Stack-Speicher als Adreß-Registersatz oder eines der internen Register als Stack-Pointer-Register für einen externen Stack-Speicher zu benützen. Darüber hinaus hat der Programmierer Zugriff zu jedem der 16-Bit-Register im Registersatz: er kann Daten von einem zum anderen Register transportieren und er kann jeweils eines davon als Befehlszähler, Stack-Pointer-Register, Indexregister usw. definieren. Diese Flexibilität ermöglicht eine vielseitige Programmierung.

Da die Register des Registersatzes 16-Bit-Format haben, können sie zusätzlich zum Abspeichern von Daten-Adressen in einem indirekten Register-Adressierschema verwendet werden. Man beachte, daß der Abruf, welches der Register der Programmschrittzähler ist, als 4-Bit-Code im P-Register gespeichert ist. Ähnlich legen die Codes in den 4-Bit-Registern N und X fest, welches der 16-Bit-Register das Stack-Pointer-Register und welches das Indexregister ist.

Grundarchitektur der PPS-8 CPU
(Bild 2.17)

Die PPS-8-CPU, einschließlich der dazugehörigen Bausteine (Bild 2.17), sind derart organisiert, daß sie hohe Ein/Ausgabe-Datenraten bewältigen können. Ein Chip dieses Systems ist eine DMA-Steuerung. Dieser Chip steuert den direkten Speicherzugriff (DMA) durch das Bereitstellen der Speicheradresse und der Interface-Signale für die Ein/Ausgabe-Steuerungen. Bei dieser Baustein-Familie handelt es sich um ein vollständiges System, was die ergänzenden Bausteine anbelangt, die im Gesamtsystem die CPU zu einem System komplettieren.

Grundarchitektur des PPS-25-Prozessors
(Bild 2.18)

Der PPS-25, skizziert in Bild 2.18, ist ein Mikroprozessor größerer Wortlänge. Die tatsächliche Genauigkeit dieses Prozessors beträgt 25 Bits. Diese Wortlänge wird erreicht durch einen 4-Bit-Addierer, der Daten in 4-Bit-Blöcken zur CPU senden kann, die dann im Register A als Resultat akkumuliert werden. Der PPS-25-Prozessor ist jedoch nicht nur eine 25-Bit-Maschine, sondern auch das Hauptprogramm-ROM ist außerhalb der CPU angeordnet. Daraus folgt, daß die Programmierung und die Herstellung des Prozessors zwei getrennte Arbeitsgänge sind.

Allgemein besitzt der PPS-25 eine spezielle Architektur, die sinnvoll ist bei Anwendungen, die eine hohe numerische Genauigkeit verlangen, wie z.B. bei wissenschaftlichen Berechnungen.

Eine andere Methode, die numerische Genauigkeit zu verbessern, ist mit einer aus 16-Bit-Prozessoren gebildeten parallelen Architektur möglich, wie sie z.B. der 9900, der 'S481 und der SBP-0400 haben. Mit diesen Parallel-Architekturen erreicht man eine gegebene Genauigkeit sehr viel schneller als mit 4- oder 8-Bit-Prozessoren. Zusätzlich bieten diese Prozessoren parallele Adressierung, Stack-Speicher-Fähigkeit und Ein/Ausgabe-Einrichtungen von echten Universal-Prozessoren.

Bei diesen Prozessoren mit größeren Wortlängen ist die Tendenz zu erkennen, die ROMs für Mikroprogramme auf dem Chip zu plazieren, um komplexere Befehle, wie Multiplikation und Division, implementieren zu können.

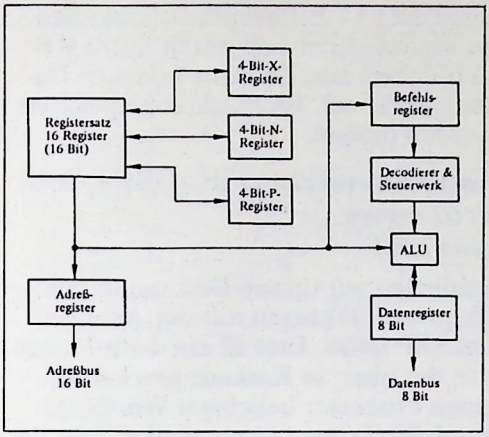


Bild 2.16
Grundarchitektur des CMOS-Prozessors

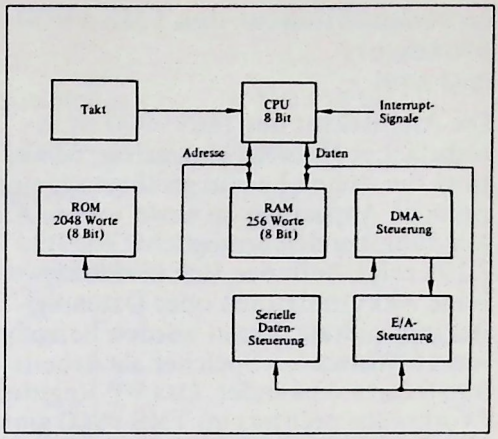


Bild 2.17
Grundarchitektur der PPS-8-CPU und der Chip-Familie

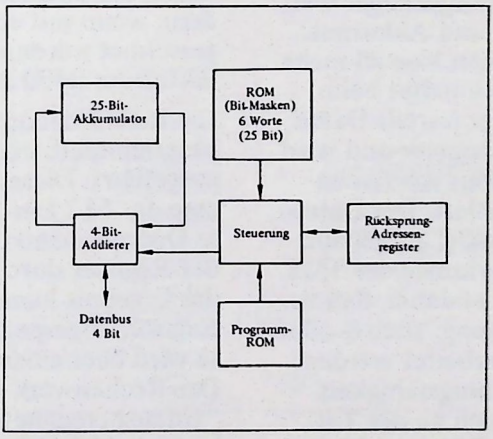


Bild 2.18
Grundarchitektur des PPS-25-Prozessors

Grundarchitektur des TMS 9900-Prozessors
(Bild 2.19)

Die Architektur des TMS 9900 ist in mehrfacher Hinsicht einzigartig, da sie, trotz der hohen Leistungsfähigkeit, eine einfache Anpassung an einen weiten Anwendungsbereich ermöglicht. Wie Bild 2.19a zeigt, befinden sich im Prozessor keine Akkumulatoren oder Datenregister mehr. Statt dessen werden Bereiche von 16 Worten im Speicher als Arbeitsregistersatz verwendet. Das WP-Register (Workspace pointer) im TMS 9900 gibt jeweils die Adresse des Registers 0 im Speicher an. Die 16 Arbeitsregister können beliebig als Daten-, Adress-, Index-Register oder Akkumulatoren verwendet werden. Dies bietet bei der Programmierung eine hohe Flexibilität und berücksichtigt die Tatsache, daß Speicher immer schneller und preisgünstiger werden, wodurch Register und Akkumulatoren im Prozessor keinen Vorteil mehr bieten. Der Speicher beinhaltet beim TMS 9900-System daher jeweils Daten, Programm und Arbeitsregister und wird über den 15-Bit-Adressbus wortweise (1 Wort = 16 Bit) adressiert. Der Datenbus ist dabei 16 Bit parallel aufgebaut. Bild 2.19b zeigt die Merkmale des TMS 9900. Von Bedeutung ist dabei, daß neben der 1-Bit-Verarbeitung, auch 8- oder 16-Bit-Datenworte verarbeitet werden können. Die hohe Leistungsfähigkeit des Befehlssatzes läßt sich an der Tatsache erkennen, daß Befehle wie Multiplikation und Division bereits fest verdrahtet sind. Die Ein-/Ausgabestruktur weist neben dem DMA-Betrieb und der parallelen 8- oder 16-Bit-Ein-/Ausgabe noch eine programmierte Ein-/Ausgabe auf. Diese programmierte Ein-/Ausgabe ermöglicht ein variables Ein-/Ausgabeformat von 1 bis 16 Bit, also z.B. 1 Bit für Steuerungszwecke, oder auch 8, 10, 12 Bit ect. für den Anschluß von A/D-Wandler oder sonstiger Peripherie. Die Interruptstruktur des TMS 9900 läßt insgesamt 17 Interruptmöglichkeiten zu und zeichnet sich durch kurze Reaktionszeit aus, da keine internen Datenregister ect. im Stack abgespeichert werden müßen.

Grundarchitektur des SBP 0400-Prozessors
(Bild 2.20)

Schließen wir unsere Diskussion der Chip-Architekturen mit der Analyse des SBP 0400. Dies ist ein 4-Bit-Prozessor, der aber, in Kaskade geschaltet, einen Prozessor beliebiger Wortlänge aus 4-Bit-Teilen (slices) ergibt. Sein Befehlsvorrat ist ähnlich der weit verbreiteten ALU SN 74181. Zusätzlich stellt die Expandierung des 0400-Prozessors auf eine beliebige Wortlänge nur ein einfaches Chip-Verbindungsproblem dar, verglichen mit dem hohen Softwareaufwand, den manche Prozessoren erfordern, wenn mit erhöhter Genauigkeit gerechnet werden soll. Die Basis-Architektur des 0400 ist in Bild 2.20 gezeigt.

Die Befehlsdecodierung wird durch ein programmierbares Logik-Array (PLA) ausgeführt. Dieses Feld selektiert irgendeine der 512 Ein-Takt-Operationen. Diese Operationen können mit irgendeinem der Register durchgeführt werden und das Ergebnis kann in irgendeinem der Register abgespeichert werden. Die Adresse wird über einen 4-Bit-Bus ausgegeben. Das Rechenwerk (ALU) arbeitet mit "vorausberechnetem Übertrag" (carry-look-ahead) und entsprechenden Übertrags-Ein- und -Ausgängen um die Kaskadierung zu größeren Wortlängen zu ermöglichen.

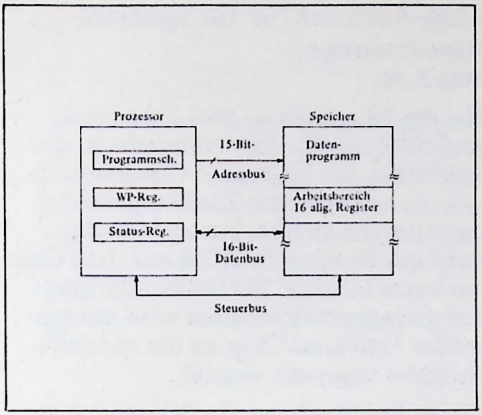


Bild 2.19a
Grundarchitektur des TMS 9900

Datenformat:	1, 8 und 16 Bit
Befehlsvorrat:	69 Befehle
Interrupt:	17 Vektor-interrupts
Adressierungs-arten:	8
E/A:	DMA, 8/16 Bit parallel 1 - 16 Bit programmiert

Bild 2.19b
Merkmale des TMS 9900

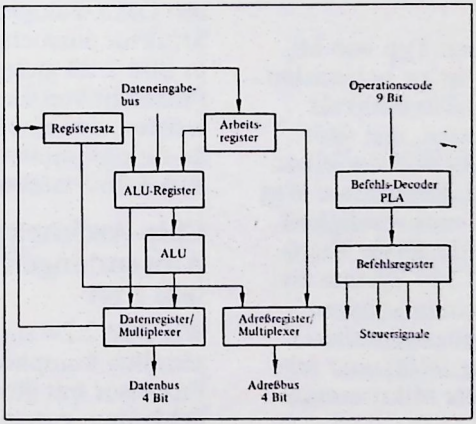


Bild 2.20
Grundarchitektur des SBP0400-Prozessors

Vier SBP 0400-Prozessoren kombiniert – 16-Bit-Parallel-Maschine
(Bild 2.21)

Durch die Kaskadierbarkeit des SBP 0400 ist es sehr einfach, einen Prozessor beliebiger Wortlänge, wie z.B. die 16-Bit-Parallel-Maschine in Bild 2.21, zu realisieren. Hierzu muß man lediglich die Übertrags-Ein- mit den Übertrags-Aus-Anschlüssen sowie die Schiebeleitungen für die Arbeitsregister miteinander verbinden. Damit ist die externe Übertrags-Generierung sichergestellt. Sodann werden Instruktions- und Eingabe-Daten in das 16-Bit-System gesendet. Dies ist nicht nur eine einfache logistische Implementierung, sondern aufgrund der I^2L-Logik besitzt diese Schaltung eine sehr hohe Arbeitsgeschwindigkeit. Wir werden I^2L detailliert in Kapitel 3 besprechen.

Es gibt zwei Wege, diesen Typ von Mikroprozessor-Architektur zu betrachten. Der eine besteht darin, den 0400 als einen Baustein aufzufassen, mit dem ein System aus Chips ein und derselben Familie aufgebaut wird. Der andere Weg ist, von der bipolaren Geschwindigkeit des 0400 auszugehen und dieses Kriterium zu benutzen, ein Chip-System zusammenzustellen, das einen anderen Mikroprozessor nachbildet (emuliert). Alles in allem bietet der 0400 eine sehr einfache Lösung für viele Mikrocomputerprobleme.

Damit ist die Diskussion über Chip-Architekturen abgeschlossen. Viele der Architekturen sind sehr ähnlich. Sie unterscheiden sich mehr in einzelnen Spezifikationen als in ihrer Grundstruktur. Eine Ausnahme ist der Unterschied zwischen der Struktur für spezielle Anwendungen und der Universal-Strukturen.

Chip-Architektur für spezielle Anwendungen
(Bild 2.22)

Wie die Übersicht in Bild 2.22 zeigt, sind für spezielle Anwendungen hoher Stückzahl, die begrenzte Geschwindigkeits- und begrenzte Ein/Ausgabe-Anforderungen stellen, ein Spezial-Baustein mit Programm-ROM auf dem Chip die beste Lösung. Für Anwendungen mit geringen Stückzahlen muß ein passender Universal-Chip an die spezielle Aufgabe angepaßt werden.

Chip-Architektur für Kommunikations-Anwendungen
(Bild 2.23)

Werden hohe Ein/Ausgabe-Daten-Raten wie in einem Kommunikations-System benötigt, sollte man einen Prozessor mit DMA-Fähigkeit und guter Interrupt-Struktur aussuchen. Wie die Übersicht in Bild 2.23 außerdem zeigt, sollte der Prozessor von einem Hersteller bezogen werden, der eine komplette Chip-Familie für die Implementierung effektiver E/A-Schnittstellen bereitstellt.

Chip-Architektur für universelle Anwendungen
(Bild 2.24)

Wie Bild 2.24 ausführt, ist im Falle einer ziemlich komplexen Anwendung ein Prozessor mit guten Ein/Ausgabe-Möglichkeiten, gutem Befehlsvorrat und einem guten unterstützenden Chip-System (für die Peripherie) zu wählen. Mikroprogrammierbarkeit sowie externer oder großer interner Stack-Speicher sind weitere wichtige Forderungen. Und schließlich sollte das System einfach zu programmieren sein.

Zusammenfassung

In dieser Lektion wurden System- und Chip-Architektur beschrieben sowie die Entwicklung und Zusammenhänge für die Leistungsfähigkeit eines Systems aufgezeigt.

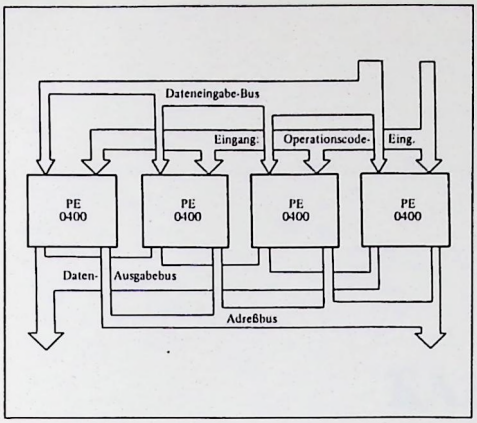


Bild 2.21
Vier SBP0400 Prozessoren, kombiniert zu einer 16-Bit Parallel Maschine

- ROM-Programmierung auf dem Chip
- Begrenzte Geschwindigkeit
- Begrenzte Ein-/Ausgabe (I/O)
- Kurze Wortlänge

Beispiele:
Druckersteuerung
Haushaltsgerätesteuerung
(z.B. Ofen ect.)

Bild 2.22
Chip-Architektur für spezielle Anwendungen

- Benötigt exzellente E/A-Bedienbarkeit
- Verlangt DMA
- Gute Interrupt-Struktur
- Chipsystem, das Interface-Chips enthält, ist vorzuziehen

Beispiele:
Verschlüssler
Datenend-Prozessoren
Dateneingabe-Terminals

Bild 2.23
Chip-Architektur für Kommunikations-Anwendungen

- Gute E/A-Möglichkeit
- Vielseitiger Befehlsvorrat
- Mikroprogramm-Stackspeicher
- Einfach zu programmieren – gute Software-Hilfen

Beispiele:
Kassenterminals
Mikrocomputer

Bild 2.24
Chip-Architektur für universelle Anwendungen

KAPITEL 3

TECHNOLOGIEN ZUR HERSTELLUNG VON MIKROPROZESSOR-CHIPS

Dieses Kapitel stellt eine grundlegende Einführung in die heute verwendeten Fertigungstechnologien dar. Darüber hinaus werden Verfahren erarbeitet, die den Entwickler in die Lage versetzen, die richtige Technologie für seine Anwendung auszuwählen.

Technologien zur Herstellung von Mikroprozessor-Chips

In diesem Kapitel werden die für die Herstellung der Mikroprozessoren verwendeten Fabrikationsprozesse diskutiert. Dies vermittelt einen Eindruck von den durch die verschiedenen Technologien bedingten Fähigkeiten und Grenzen der Mikroprozessoren. Deshalb wird hier dargelegt, welche Herstellungsprozesse verwendet werden und ihre prinzipiellen Vor- und Nachteile.

Merkmale der Mikroprozessor-Chips
(Bild 3.1)

Bild 3.1 zeigt einige Anforderungen an Mikroprozessoren-Technologien. Diese Technologien müssen gewährleisten, daß man eine Integrationsdichte von 2000 Gattern pro Chip erreichen kann, wie es für einen typischen 4-Bit-Mikroprozessor erforderlich ist. Die gewählte Technologie muß daher eine effektive Ausnutzung des Siliziummaterials gewährleisten. Die Einfachheit des Herstellungsprozesses muß sich durch eine geringe Anzahl von Foto-Masken-Schritten auszeichnen, um bei der Herstellung hohe Ausbeuten und Wirtschaftlichkeit zu sichern. Bei dem Aufbau eines Mikroprozessors sollten nur solche hochintegrierten Bausteine verwendet werden, die die Anforderungen des Mikroprozessors erfüllen.

Technologien für Mikroprozessor-Chips
(Bild 3.2)

Es sind beim Entwurf und der Herstellung von Halbleiter-Chips für logische Schaltungen mehrere Dinge zu beachten. Daher behandelt dieses Kapitel mehrere für Logikschaltungen verwendete Technologien (Bild 3.2). Dazu gehören der bipolare Transistor mit und ohne Schottky-Diode, die MOS- und die I^2L-Technologie.

Querschnitt durch einen bipolaren Transistor
(Bild 3.3)

Bild 3.3 beginnt den Rückblick auf die Entwicklung der Halbleiter auf dem Gebiet der digitalen Schaltkreise mit Rückblick auf den bipolaren Transistor. Dieser besteht aus zwei dicht benachbarten PN-Übergängen. Bei einem NPN-Transistor ist der Emitter/Basis-Übergang mittels einer Spannungsquelle in Durchlaßrichtung vorgespannt, wodurch Elektronen in die Basis-Region eingespeist werden. Dabei entsteht dort ein Überschuß an Minoritätsträgern. Diese diffundieren zu dem Kollektor-Übergang, wo sie von dem elektrischen Feld des Gebietes mit positiver Raumladung am in Sperr-Richtung vorgespannten Kollektor/Basis-Übergang in den Kollektor hinein beschleunigt werden. Ein kleiner Teil der Elektronen, die das Basisgebiet überqueren, gehen durch Rekombination verloren. Deshalb muß ein Basisstrom von außen angelegt werden, damit genügend Löcher für den Rekombinationsprozeß zur Verfügung stehen.

- Mindestens 2000 Gatter pro Chip
- Effektive Ausnutzung des Siliziummaterials
- Einfacher Herstellungsprozeß
- Die elektrischen Anforderungen müssen die Leistungsfähigkeit eines Mikroprozessors gewährleisten

Bild 3.1
Anforderungen an Mikroprozessor-Technologien

- Bipolar
- Bipolar mit Schottky-Diode
- MOS
- I^2L

Bild 3.2
Technologien für Mikroprozessor-Chips

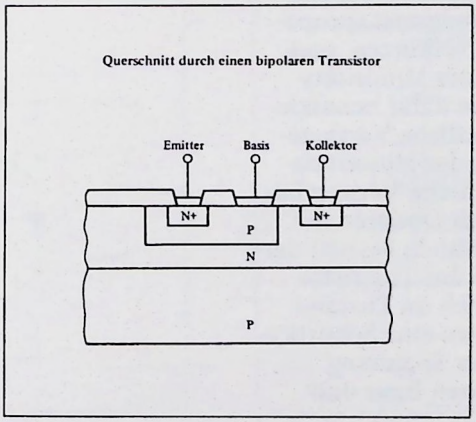


Bild 3.3
Querschnitt durch einen bipolaren Transistor

Schaltsymbol für den bipolaren Transistor
(Bild 3.4)

Der Transistor weist eine Stromverstärkung auf. Sie ist durch β definiert und gleich dem Verhältnis aus Kollektorstrom zu Basisstrom (Bild 3.4).

Logisches Schalten eines bipolaren Transistors
(Bild 3.5)

Die logischen Zustände 0 und 1 sind in Bild 3.5 durch die Schaltzustände EIN und AUS des Transistors dargestellt. Ein bipolarer Transistor kann von AUS nach EIN geschaltet werden, indem man einen ausreichenden Basisstrom fließen läßt, wodurch der Transistor in den Sättigungsbereich, d.h. in den Zustand EIN übergeht. Daher ergibt sich eine Verzögerungszeit, wenn der Transistor in den Zustand AUS zurück geführt werden soll. Um die Verzögerungszeit (propagation delay time) zu verkürzen, muß man die Speicherzeit der Minoritätsträger reduzieren. Eine dafür benutzte Methode ist die wesentliche Verringerung der Lebensdauer der Minoritätsträger durch das künstliche Verunreinigen des Siliziums durch Dotieren mit Gold. Eine andere Methode besteht darin, zu verhindern, daß der Transistor in den Sättigungsbereich im Zustand EIN gelangt, indem man eine Schottky-Diode zum Kappen der Spannung (voltage clamp) zwischen Basis und Kollektor schaltet.

TTL-NAND-Gatter und dessen Wahrheitstabelle
(Bild 3.6)

Bild 3.6 zeigt ein typisches NAND-Gatter. Ein einziger Transistor T1 mit mehrfachem Emitter wird für die Eingänge verwendet. Jede Emitter-Basis-Diode dient als ein Eingang, während die Basis/Kollektor-Diode für alle Eingänge in Reihe liegt. Im Ausgangsteil wird ein aktiver Pull-up-Transistor T4 verwendet, der die nötige Stromverstärkung für das Schalten in beiden Richtungen zur Verfügung stellt. Der Schaltzustand von T1 dient dazu, denjenigen von T2 zu steuern, um den Ausgang bei T3 und T4 festzulegen. Eine "Null" an einem oder an mehreren der Eingänge A, B und C bewirkt, daß T1 leitet, T2 sperrt und daraufhin auch T3 sperrt, wodurch man am Ausgang das Ergebnis "Eins" erhält, d.h. daß am Ausgang ein hohes Spannungspotential liegt. Ist A, B und C "Eins", sperrt T1, während T2 und T3 leiten und am Ausgang daraufhin ein tiefes Spannungspotential liegt, d.h. auf "Null" ist. Bild 3.6b zeigt die Wahrheitstabelle eines NAND-Gatters. Eine "logische Null" an irgendeinem Eingang bewirkt eine "logische Eins" am Ausgang und eine "Eins" an allen Eingängen hat eine "Null" am Ausgang zur Folge.

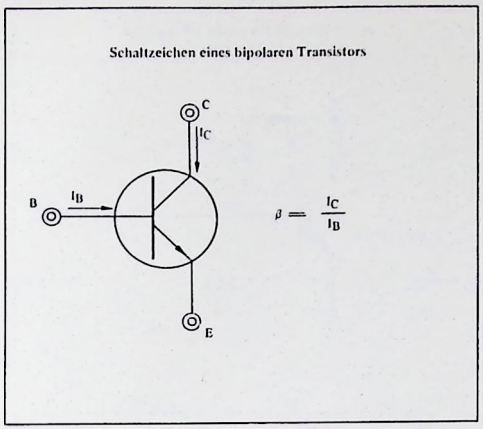


Bild 3.4
Schaltsymbol eines bipolaren Transistors

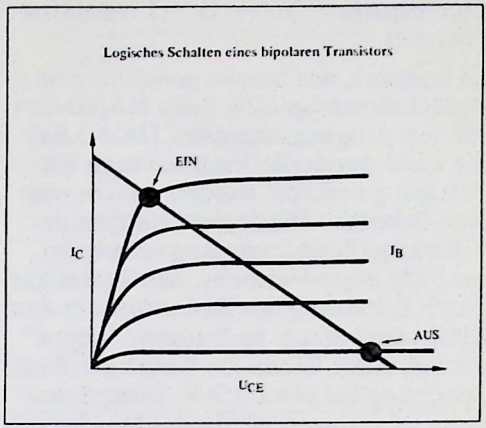
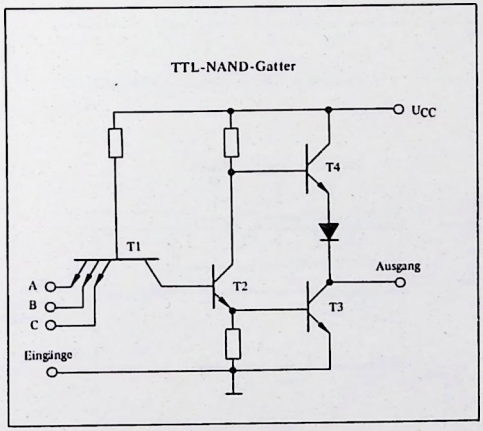


Bild 3.5
Logisches Schalten eines bipolaren Transistors



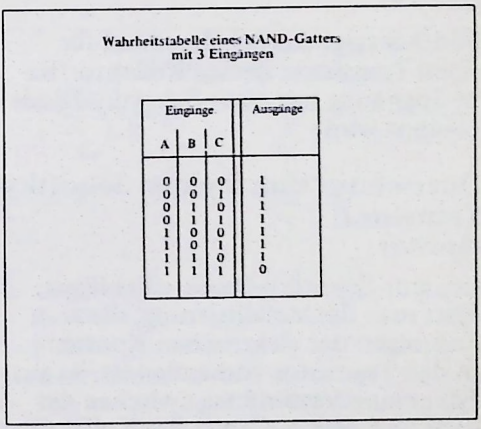


Bild 3.6
TTL-NAND-Gatter und zugehörige Wahrheitstabelle

Der bipolare Schottky-Transistor
(Bild 3.7)

Es existiert, wie bereits erwähnt, eine Ausschaltverzögerung beim NAND-Gatter am Ausgangstransistor (Bild 3.6a). Sie kann durch die Verhinderung der Sättigung verkürzt werden, indem man eine Schottky-Diode zum Kappen der Kollektor/Basis-Spannung verwendet. Im Falle eines Versuchs, den Transistor durch Erhöhung des Basisstroms in den Sättigungsbereich zu bringen, "kappt" die Schottky-Diode die Kollektor/Basis-Spannung bei etwa 0,3 V. Diese Spannung ist geringer, als für das Durchschalten eines PN-Überganges benötigt wird. Auf diese Weise wird die Sättigung verhindert.

Schaltsymbol für den Schottky-Transistor
(Bild 3.8)

Bild 3.8 zeigt das Schaltsymbol für einen Transistor, dessen Kollektor/Basis-Spannung mit einer Schottky-Diode gekappt wird.

Querschnitt durch einen Schottky-Transistor
(Bild 3.9)

Um eine Schottky-Diode einzufügen, führt man die Metallisierung, die zum Anbringen der elektrischen Kontakte an den Transistor vorhanden ist, so aus, daß er eine Verbindung zwischen der Basis und dem N-Gebiet des Kollektors herstellt. Die Gestaltung eines Transistors mit Schottky-Diode nach Bild 3.9 stimmt völlig mit der üblichen TTL-Technologie für integrierte Schaltungen überein.

Schottky-TTL-NAND-Gatter
(Bild 3.10)

Ein einfaches NAND-Gatter mit Schottky-Dioden ist in Bild 3.10 dargestellt. Durch das Kappen der Kollektor/Basis-Spannung werden höhere Schaltgeschwindigkeiten erreicht.

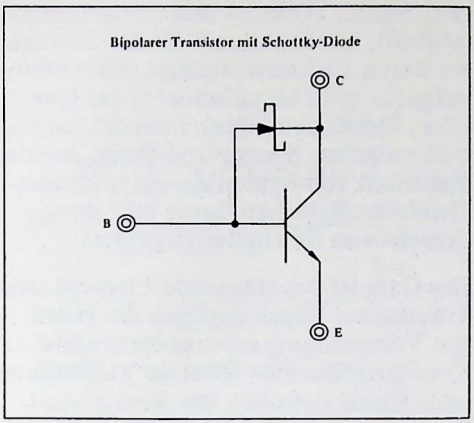


Bild 3.7
Der bipolare Schottky-Transistor mit Schottky-Diode

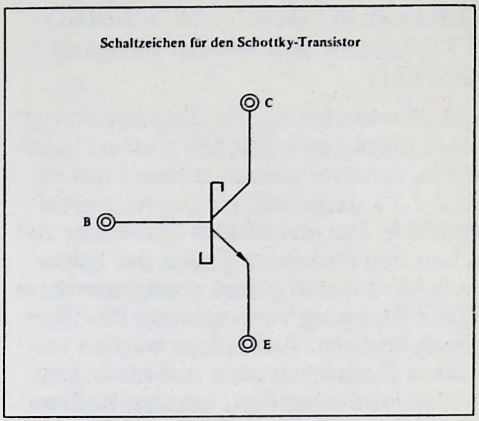


Bild 3.8
Schaltsymbol für den Schottky-Transistor

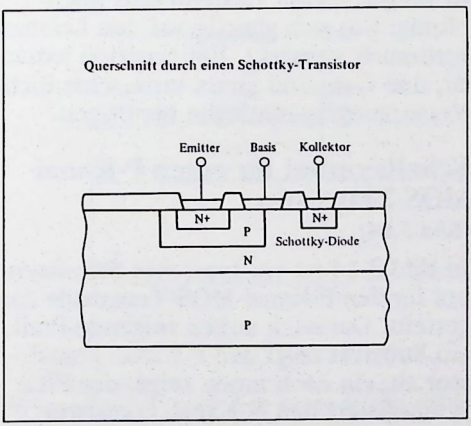


Bild 3.9
Querschnitt durch einen Schottky-Transistor

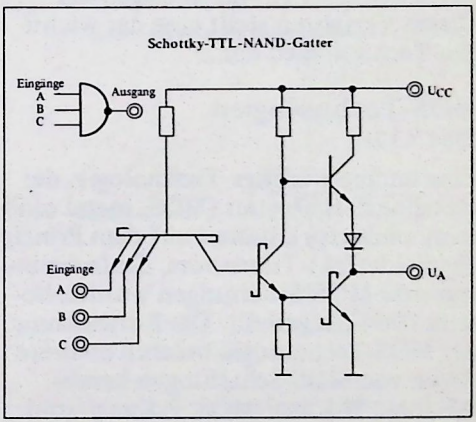


Bild 3.10
Schottky-TTL-NAND-Gatter

Querschnitt durch ein Schottky-TTL-Gatter mit einem Eingang
(Bild 3.11)

Einige wichtige der für die Herstellung eines bipolaren logischen Gatters benötigten Schritte werden anhand des in Bild 3.11 dargestellten Querschnittes deutlich. Die elektrische Isolierung zwischen den einzelnen Teilen der Schaltung wird durch genau positionierte, in Sperr-Richtung vorgespannte PN-Übergänge erreicht. Außerdem werden versenkte Kollektorzonen mit niedrigem Widerstand eingefügt, um den Reihenwiderstand der Kollektoren zu verringern und kurze Verzögerungszeiten zu erhalten. Last- und Vorspannungswiderstände werden mit isolierten P-dotierten Zonen erzeugt, die gleichzeitig mit dem P-Gebiet der Basis hergestellt werden. Auf diese Weise stellt man die bekannten Schottky-TTL-Schaltungen her. Dieses Verfahren stellt eine der wichtigen Technologien dar..

MOS-Technologien
(Bild 3.12)

Eine andere wichtige Technologie, der Metalloxid-Halbleiter (MOS, metal oxide semiconductor), basiert auf dem Prinzip des Feldeffekt-Transistors. Einfache integrierte MOS-Schaltungen wurden bereits 1964 hergestellt. Die Entwicklung der MOS-Technologie brachte mehrere Arten von MOS-Schaltungen hervor: P-Kanal-, N-Kanal-MOS, P-Kanal- und N-Kanal-MOS zusammen als CMOS (complementary MOS) und SOS-Bausteine (silicon on saphire) (Bild 3.12).

P-Kanal-MOS
(Bild 3.13)

Der Querschnitt durch einen P-Kanal-MOS-Transistor (Metall-Gate) ist in Bild 3.13 dargestellt. Ein dünnes, leicht N-dotiertes Siliziumplättchen bildet das Grundmaterial dieses Transistors. Zwei dicht benachbarte, stark P-dotierte Zonen, Source (Quelle) und Drain (Senke, Abfluß), werden innerhalb des Substrates durch Diffusion erzeugt. Eine dünne aufgedampfte Metallschicht, das Gate (Tor, Steuerelektrode), bedeckt das Gebiet zwischen Source und Drain, ist aber elektrisch von beiden Zonen und dem Halbleiterplättchen durch eine dünne Schicht von Siliziumoxid isoliert.

Das Gate ist das steuernde Element des Transistors. Durch Anlegen der richtigen Vorspannung erzeugt diese Gate-Konfiguration eine leitende Verbindung bzw. Kanal zwischen der Source- und Drainzone (Bild 3.13). Der kleinste Wert der Gate-Vorspannung, der benötigt wird, um den Kanal zu formieren, wird als Gate-Schwellspannung UGT (gate threshold voltage) bezeichnet und ist von dem jeweiligen Herstellungsverfahren abhängig. Der P-Kanal-Transistor ist im gesperrten Zustand sehr hochohmig, was sich günstig auf den Leistungsverbrauch auswirkt. Ein Nachteil jedoch ist, daß Gate und Drain unterschiedliche Versorgungsspannungen benötigen.

Schaltsymbol für einen P-Kanal-MOS-Transistor
(Bild 3.14)

In Bild 3.14 ist ein typisches Schaltsymbol für den P-Kanal-MOS-Transistor dargestellt. Der nach außen zeigende Pfeil am Substrat zeigt den P-Kanal-Transistor an; ein nach innen zeigender Pfeil symbolisiert den N-Kanal-Transistor. Die unterbrochene Linie symbolisiert den Anreicherungstyp (selbstsperrend). Wäre sie durchgezogen, symbolisierte dies einen Verarmungstyp (selbstleitend).

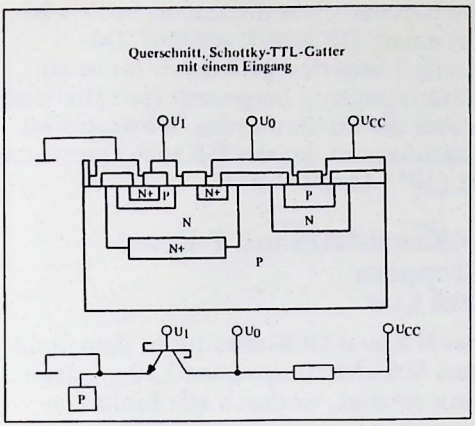


Bild 3.11
Querschnitt durch ein Schottky-TTL-Gatter mit einem Eingang

- P-Kanal
- N-Kanal
- CMOS
- SOS

Bild 3.12
MOS-Prozesse

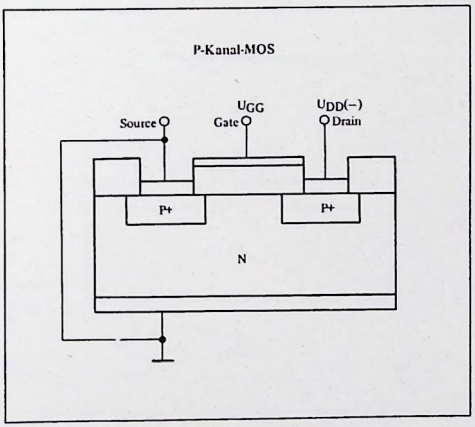


Bild 3.13
P-Kanal-MOS-Transistor

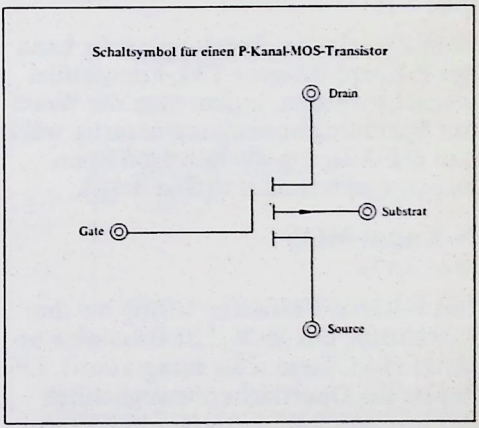


Bild 3.14
Schaltsymbol für einen P-Kanal-MOS-Transistor

Der P-Kanal-Anreicherungstyp als Inverter
(Bild 3.15)

Eine typische, aus P-Kanal-Transistoren zusammengesetzte Schaltung stellt der in Bild 3.15 abgebildete Inverter dar. Die Versorgungsspannungen betragen −12 V für UGG und −6 V für UDD. Der Transistor T1 befindet sich im Zustand EIN, während T2 durch das Eingangssignal gesteuert wird. Falls der Eingang auf log. "1" liegt, befindet sich T2 im Zustand AUS und der Ausgang liegt etwa auf UDD, d.h. er liegt auf log. "0" und zwar auf einer negativen Spannung. Geht der Eingang auf log. "0" und schaltet damit T2 in den Zustand EIN, dann geht der Ausgang auf log. "1", d.h. etwa auf das Massepotential.

TTL-kompatibler MOS-Inverter
(Bild 3.16)

Zum Zwecke der Pegelanpassung kann der P-Kanal-Inverter TTL-kompatibel gemacht werden, indem man die Werte der Spannungsversorgung derartig wählt, daß der Ausgang die richtigen Spannungspegel bekommt (Bild 3.16).

N-Kanal-MOS
(Bild 3.17)

Das P-Kanal-Verfahren wurde bei der Einführung der MOS/LSI-Bausteine benutzt (LSI, large scale integration). Leider ist die Oberflächenbeweglichkeit der inversen Ladungsträger bei diesem Verfahren relativ gering, was eine niedrige Geschwindigkeit des Bausteins zur Folge hat. Höhere Trägerbeweglichkeiten kann man durch einen anderen MOS-Prozeß, den N-Kanal-Prozeß, erreichen (Bild 3.17). Der N-Kanal-MOS-Feldeffekt-Transistor kann dadurch hergestellt werden, daß man die N-dotierten Source- und Drainzonen durch Diffusion von Phosphor in ein P-dotiertes Grundmaterial erzeugt. Die Beweglichkeit der Ladungsträger in der Inversionsschicht ist bei N-Kanal etwa dreimal so hoch wie bei einem P-Kanal-Transistor. Der N-Kanal-Transistor wird daher meist als Verarmungstyp hergestellt (selbstleitend), wobei die Dotierung des Substrates vergleichbar mit der des P-Kanal-Transistors ist (10^{15} Atome/cm^3).

N-Kanal-MOS mit P-Kanal-Stoppern
(Bild 3.18)

Bei N-Kanal-MOS wird unter dem Oxid eine Schicht mit inversen Ladungsträgern erzeugt, wodurch alle Einheiten kurzgeschlossen werden. Dies kann korrigiert werden, indem man durch Diffusion oder Ionenimplantation erzeugte P-Kanal-Stopper ("Isolierschichten") an den entsprechenden Stellen einfügt.

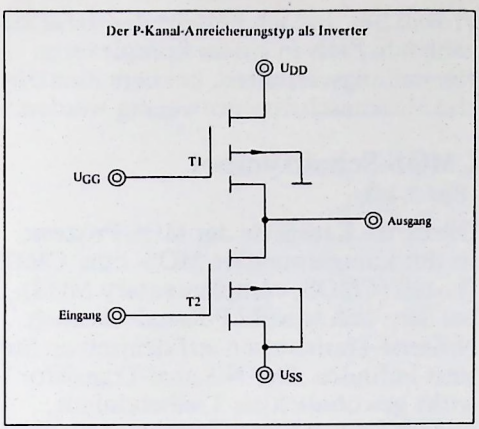


Bild 3.15
Der P-Kanal-Anreicherungstyp als Inverter

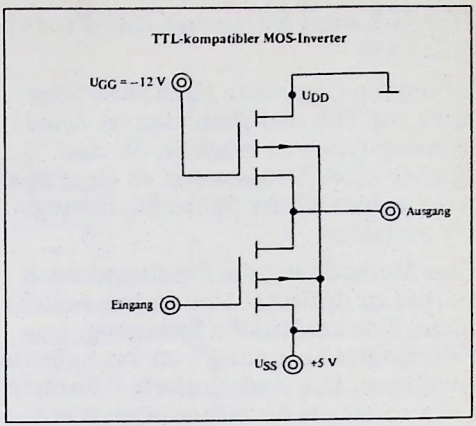


Bild 3.16
TTL-kompatibler MOS-Inverter

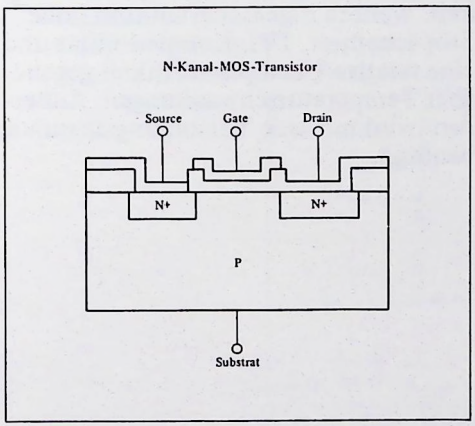


Bild 3.17
N-Kanal-MOS-Transistor

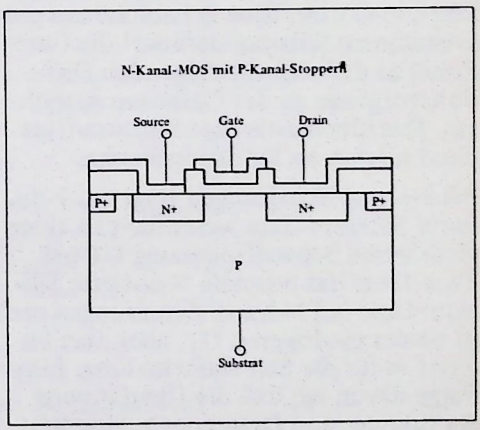


Bild 3.18
N-Kanal-MOS mit P-Kanal-Stoppern

Der N-Kanal-Silizium-Gate-Prozeß
(Bild 3.19)

Es werden besondere Methoden eingesetzt, um N-Kanal-Halbleiter zu Anreicherungstypen zu machen, die das Fließen eines Stromes erst ab einer Spannung größer als der Schwellspannung U_T gestatten.

Eine Methode ist, das P-Substrat noch stärker zu dotieren; eine andere besteht darin, eine zusätzliche Spannung, eine "Rückwärts-Spannung", an das Substrat anzulegen. Das stark dotierte P-Substrat kann zu einem Anreicherungstyp mit niedriger Schwellspannung U_T führen, der bei völliger TTL-Kompatibilität mit einer einzigen Spannungsquelle von +5 V auskommt.

Bild 3.19 zeigt ein selbstausgerichtetes Silizium-Gatter. Eine Schablone aus polykristallinem Silizium definiert die Gate-Zone, so daß sich die folgenden Diffusionsvorgänge an der Gatezone ausrichten. Das hitzebeständige Silizium-Gate dient hierbei als Diffusionsmaske.

Bei P-Kanal-Schaltungen trägt das P-dotierte Silizium-Gate wesentlich zu einer niedrigeren Schwellspannung U_T bei. Zwar führt das normale N-dotierte Silizium-Gate bei N-Kanal-Schaltungen nicht zu einem niedrigeren U_T, aber man benutzt es für die Selbstausrichtung. Eine Folge davon ist, daß die Überlappung der Source- und Drainzonen minimiert und die Kapazität verringert wird. Dies führt zu einer erhöhten Arbeitsgeschwindigkeit der Bausteine.

Dieser Silizium-Gate-Prozeß ermöglicht zusätzlich neben den Diffusionstunneln und der üblichen Verbindungs-Metallisierung eine weitere (zweite) Metallisierung einzuführen.

Alle genannten Faktoren tragen zu einer rund 5fachen Vergrößerung der Geschwindigkeit und Integrationsdichte im Vergleich zum P-Kanal-Aluminium-Gate-Prozeß bei. Jedoch besteht der dafür zu zahlende Preis in einem komplexeren Herstellungsverfahren, bei dem zusätzliche Maskenschritte notwendig werden.

CMOS-Schaltsymbol
(Bild 3.20)

Die dritte Kategorie der MOS-Prozesse ist der komplementäre MOS- oder CMOS-Prozeß (CMOS, complementary MOS), bei dem sich sowohl P-Kanal- als auch N-Kanal-Transistoren auf demselben Substrat befinden. Der N-Kanal-Transistor wirkt gewöhnlich als Treibereinheit, während der P-Kanal-Transistor die Last darstellt. In einem eingeschwungenen logischen Zustand befindet sich nur ein Transistor im leitenden Zustand. Diese Anordnung führt zu einem extrem niedrigen Leistungsbedarf im Ruhezustand und zu einer hohen Arbeitsgeschwindigkeit. Weitere Eigenschaften sind hohe Störsicherheit, TTL-Kompatibilität und eine relative Unempfindlichkeit gegenüber Temperaturschwankungen. Außerdem wird nur eine Versorgungsspannung benötigt.

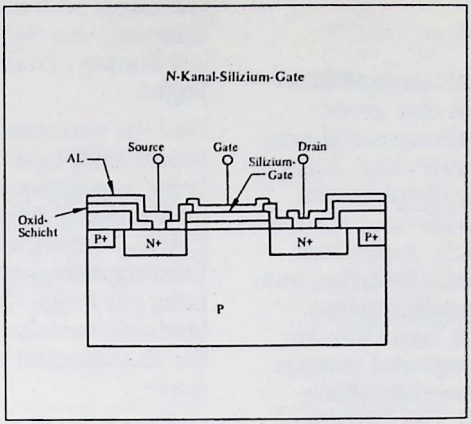


Bild 3.19
N-Kanal-Silizium-Gate-MOS-Transistor

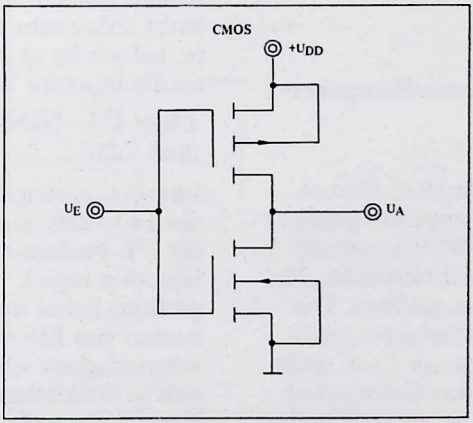


Bild 3.20
CMOS-Schaltsymbol

CMOS-Aufbau
(Bild 3.21)

Die günstigen Eigenschaften der CMOS-Halbleiter werden durch eine große Komplexität des Herstellungsverfahrens für diese Bausteine erkauft (Bild 3.21). Im Vergleich zum P-Kanal-Verfahren sind zusätzliche Diffusions- und Maskierungsschritte erforderlich. Außerdem verlangt CMOS elektrische Isolation zwischen den N- und P-Kanal-Einheiten. Deshalb sind die Source- und Drainzonen nicht zusammengelegt und müssen später getrennt durch eine Metallisierung verbunden werden. Die Erfordernisse zur Isolation und die zusätzlichen Verbindungen haben eine weitere Vergrößerung der benötigten Fläche auf dem Siliziumplättchen zur Folge, was einen weiteren Nachteil der CMOS-Technologie darstellt.

Ablauf des SOS-Herstellungsverfahrens
(Biid 3.22)

Der letzte zu nennende MOS-Prozeß wird SOS (silicon on sapphire) genannt (Bild 3.22). Die Schaltkreiskonstruktion wird auf einem nichtleitenden Material wie z.B. Saphir ausgeführt. Das Grundmaterial für das Substrat erhält man, indem man eine etwa 1 μm dicke einkristalline Schicht aus N-dotiertem Silizium auf eine einkristalline Scheibe aus Saphir epitaxial aufwachsen läßt. Diese Schicht wird daraufhin in der Form geätzt, daß isolierte Inseln für die einzelnen Einheiten entstehen. Diejenigen Inseln, aus denen N-Kanal-Einheiten entstehen sollen, werden zunächst von N-Dotierung auf P-Dotierung umgewandelt. Der nächste Schritt besteht aus der Eindiffundierung der P^+- und N^+-Source- und Drainzonen. Das Oxid an der Gatezone wird teilweise entfernt und man läßt die dünne Oxidschicht für das Gate neu anwachsen. Der letzte Prozeßschritt ist das Aufbringen der Metallisierung, was die leitende Verbindung zur Source-, Drain- und Gate-Region ergibt.

Weil die einzelnen Einheiten völlig getrennt sind, besteht zwischen ihnen keine kapazitive Koppelung. Dies hat im Vergleich zur konventionellen CMOS-Technik eine Verbesserung der Leistungsfähigkeit um eine Größenordnung zur Folge. Die Nachteile dieser Methode bestehen im wesentlichen aus der Komplexität des Herstellungsverfahrens.

Vergleicht man die bipolare TTL- mit der MOS-Technologie, dann sieht man, daß die TTL-Technologie nicht nur sehr schnelle, sondern auch recht billige Bausteine liefert. Sie benötigt allerdings mehr Siliziumfläche als MOS. MOS erfordert einen minimalen Platz und erlaubt daher sehr hohe Integrationsdichte, jedoch ist es langsamer und billiger als die bipolare Technologie.

Zwei I^2L-NOR-Gatter
(Bild 3.23)

Eine Neuentwicklung auf dem Gebiet des Entwurfs von Mikroprozessoren ist die I^2L-Technologie (I^2L, integrated injection logic). I^2L verbindet die Integrationsdichte und geringen Herstellungskosten von PMOS mit der hohen Geschwindigkeit von TTL, denn es handelt sich in Wirklichkeit um einen bipolaren Prozeß. Bemerkenswert ist, daß in dieser Technologie die Leistung- und Platzverbrauchenden Lastwiderstände von TTL fehlen. Bild 3.23 zeigt, daß I^2L ein Gatter auf ein einziges komplementäres Paar von Transistoren reduziert. In diesem zweifachen NOR-Gatter arbeitet ein vertikaler NPN-Transistor mit Multikollektoren als Inverter und ein lateraler PNP-Transistor dient als Stromquelle. Auf diese Weise sind die normalerweise bei TTL verwendeten ohmschen Lastwiderstände vermieden worden.

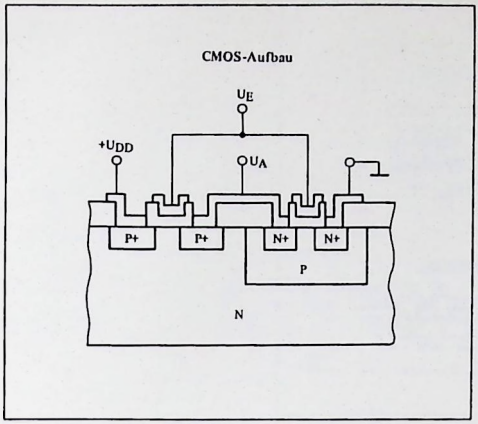


Bild 3.21
CMOS-Aufbau

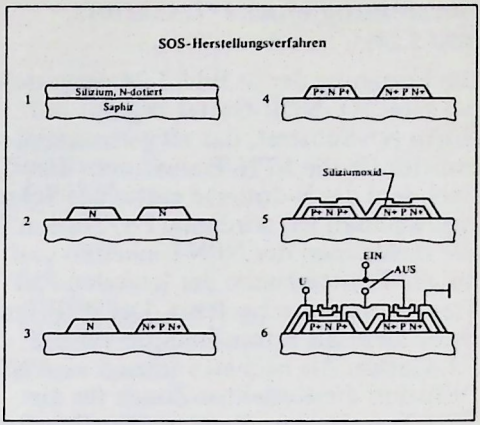


Bild 3.22
SOS-Herstellungsverfahren

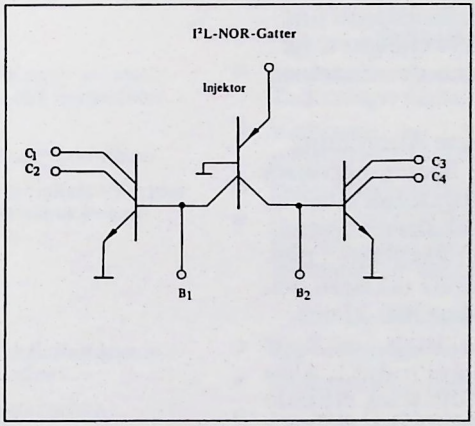


Bild 3.23
Zwei I^2L-NOR-Gatter

Herstellung eines I^2L-Gatters
(Bild 3.24)

Die Fertigung der in Bild 3.24 dargestellten zwei I^2L-NOR-Gatter beginnt mit einem N^+-Substrat, das als gemeinsamer Emitter für die NPN-Transistoren dient. Nachdem die N-dotierte epitaxiale Schicht angewachsen ist, wird eine P-Diffusion für alle Basiszonen der NPN-Einheiten und für alle Emitterzonen der lateralen PNP-Transistoren durchgeführt. Der PNP-Transistor dient als Strom-Injektor für das I^2L-Gatter. Als nächstes erzeugt eine N^+-Diffusion die Kollektor-Zonen für die NPN-Transistoren. Zwei weitere Schritte werden für die Schaffung der Kontaktlöcher und für die Metallisierung benötigt. Ist das I^2L-Gatter auf einem Siliziumplättchen angeordnet, dann benötigen beide Schaltkreiselemente nur den Platz eines einzigen Transistors. Ferner ist der Schaltungsaufbau selbstisolierend und deshalb entfällt die Notwendigkeit für eine zusätzliche Isolation der einzelnen Elemente.

Außerdem benötigt diese Anordnung der I^2L-Bausteine eine Spannungsversorgung, die dem Spannungsabfall einer in Durchflußrichtung geschalteten Si-Diode entspricht. Dies und das Fehlen der großflächigen Widerstände erklären den niedrigen Leistungsbedarf, die kleine Gatterfläche und das günstige Geschwindigkeits/Verlustleistungs-Produkt. Alles zusammen ermöglicht die stark erhöhte Integrationsdichte.

Zusammenfassung der Technologien
(Bild 3.25)

Bild 3.25 gibt eine Übersicht über die in dieser Lektion behandelten Technologien: Schottky-TTL, vier MOS-Prozesse (und zwar P-Kanal, N-Kanal, CMOS und SOS) und die bipolare I^2L-Technologie.

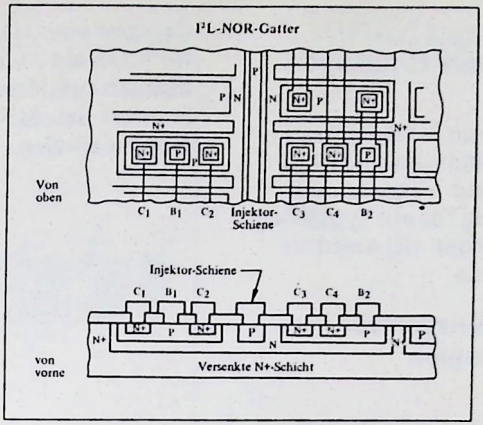


Bild 3.24
Herstellung eines I²L-Gatters

CMOS

- Verwendet sowohl P-Kanal- als auch N-Kanal-Transistoren auf demselben Substrat
- Komplexeres Herstellungsverfahren
- Verbessertes Geschwindigkeits-Verlustleistungs-Produkt gegenüber P- und N-Kanal-MOS

SOS

- Ähnlich wie CMOS
- Einheiten sind auf nichtleitendem Substrat aus Saphir angeordnet
- Verringerte Koppelkapazitäten
- Erhöhte Geschwindigkeit
- Teurer

I²L

- Vermeidet die bei TTL benötigten Lastwiderstände
- Verringerter Leistungsbedarf gegenüber bipolarer Technik
- Verbindet die Geschwindigkeit der bipolaren Technik mit der Integrationsdichte der MOS-Technik

Schottky-TTL

- Spielte eine dominante Rolle in der Technologie digitaler Schaltungen
- Verbraucht nur 1/5 der Leistung eines gesättigten TTL-Transistors
- Hohe Schaltgeschwindigkeiten
- Großer Silizium-Flächenbedarf – in Mehrchip-Prozessoren bei Mikroprozessoranwendungen organisiert

P-Kanal-MOS

- Anreicherungsbetrieb
- Höchste Integrationsdichte
- Niedrige Geschwindigkeit

N-Kanal-MOS

- Schneller als P-Kanal-MOS
- Normalerweise Verarmungsbetrieb
- Benötigt Kanalstopper zur Isolation
- Selbstausrichtendes Silizium-Gate – verbessert die elektrischen Eigenschaften
- Hohe Integrationsdichte

Bild 3.25
Technologie-Übersicht

Flächenvergleich eines NAND/NOR-Gatters mit vier Eingängen
(Bild 3.26)

Um einen Eindruck vom Verhältnis dieder Technologien zueinander zu bekommen, ist in Bild 3.26 ein Vergleich der Flächen dargestellt, die für ein typisches NAND- oder NOR-Gatter mit vier Eingängen benötigt werden.

Verzögerungszeit-Kurven für verschiedene Technologien
(Bild 3.27)

In Bild 3.27 sind die Verlustleistungs-Verzögerungszeit-Kurven für Gatter verschiedener Technologien dargestellt. I^2L liegt sehr günstig bezüglich eines niedrigen Geschwindigkeits-Verlustleistungs-Produktes.

Anwendung von Mikroprozessoren in Abhängigkeit von der Technologie
(Bild 3.28)

Die Erkenntnisse aus den Verlustleistungs-Verzögerungszeit-Kurven werden genutzt, um die Anwendungsgebiete für Mikroprozessoren bezüglich der Technologie zu klassifizieren. Man beachte den Bereich, den Schottky-TTL auf dem Gebiet der Systeme mit höchsten Anforderungen einnimmt.

Zusammenfassung

Es gibt noch weitere, in Laboratorien entwickelte Technologien, die einen Einfluß auf die Entwicklung von Mikroprozessoren nehmen könnten. Jedoch ermöglichen bereits die heute zur Verfügung stehenden Technologien ohne weiteres die Produktion von Mikroprozessoren. Schottky-TTL erlaubt Mehrchip-Prozessoren höchster Leistungsfähigkeit. MOS/LSI-Technologien sind ideal wegen ihrer höheren Integrationsdichte und geringen Leistungsverbrauchs für Ein-Chip-Mikroprozessor/Mikrocomputer. Die bipolare I^2L-Technologie ist dagegen eine starke Konkurrenz sowohl für MOS als auch für TTL und erscheint deshalb prädestiniert, eine führende Rolle unter den Mikroprozessor-Technologien zu spielen.

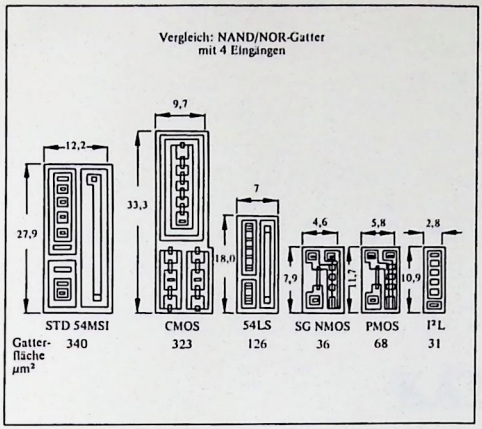


Bild 3.26
Vergleich für das NAND/NOR-Gatter mit vier Eingängen

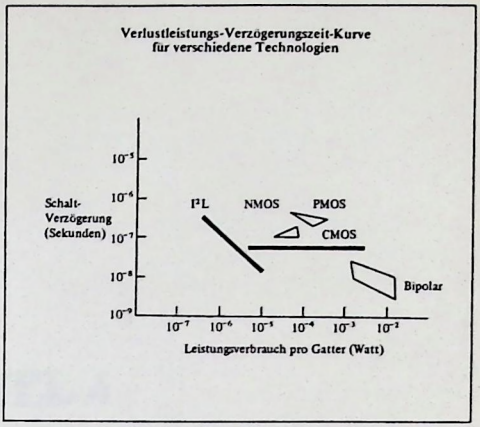


Bild 3.27
Verlustleistungs-Verzögerungszeit-Kurven für verschiedene Technologien

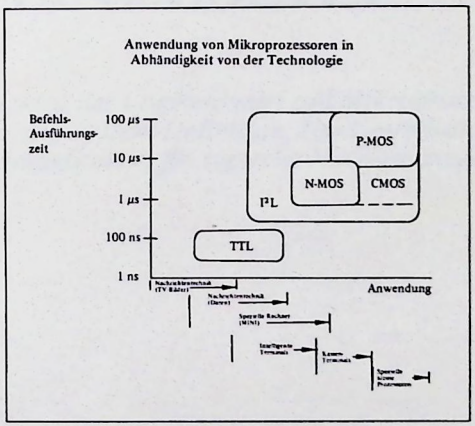


Bild 3.28
Anwendung von Mikroprozessoren in Abhängigkeit von der Technologie

KAPITEL 4

WIE IST EIN MIKROPROZESSOR EINZUSETZEN?

In diesem Kapitel wird der Logikentwurf mit Mikroprozessoren besprochen. Außerdem werden die System-Definition, Hardware-Kategorien sowie die Software-Programmdefinition für typische Mikroprozessor-Anwendungen beschrieben.

Wie ist ein Mikroprozessor einzusetzen?

Dieses Kapitel gibt eine Übersicht, wie ein Mikroprozessor nützlich eingesetzt werden kann. Zunächst werden die Fachausdrücke eingeführt und Parameter definiert, die als Richtlinien für die Frage "Wann sollte ein Mikroprozessor verwendet werden? " dienen können. Als nächstes wird umrissen, wie ein Mikroprozessor-Projekt im Zusammenhang mit den Hardware- und Software-Anforderungen bestimmt wird. Wie man für ein gegebenes System Mikroprozessoren auswählt, wird ebenfalls diskutiert.

Mikroprozessor-Fachsprache
(Bild 4.1)

Wie vorher definiert, besteht ein Mikroprozessor aus einem (oder mehreren) Baustein(en), der Funktionen ähnlich einem Minicomputer ausführt. Er ist nur langsamer und hat kürzere Wortlängen. Bild 4.1 zeigt, daß ein monolithischer Mikroprozessor ein Ein-Chip-Mikrocomputer ist, der Speicher, Rechenwerk (ALU), Leitwerk und Ein/Ausgänge enthält.

Ein anderer Ausdruck, der hier eingeführt werden muß, ist Mikroprogrammierung. Ein mikroprogrammierbarer Prozessor erlaubt dem Benutzer auf dem fundamentalen Register-Transfer-Niveau zu operieren. Beim Mikroprogrammieren arbeitet der Benutzer mit Mikrobefehlen, mit denen ein anwender-spezifischer Befehlsvorrat aufgebaut werden kann, d.h. bei mikroprogrammierbaren Prozessoren ist der Befehlsvorrat nicht fest vorgegeben, sondern er kann vom Anwender definiert werden.

Die Mikrobefehle (Basisoperationen) auf dem Register-Transfer-Niveau sind die Elemente, aus denen sich ein typischer Befehl ("Makro-Instruktion") eines festen Befehlsvorrates zusammensetzt. Per Definition besteht ein fest vorgegebener Befehlsvorrat aus "Makro-Instruktionen" kurz Befehlen und jeder Befehl aus mehreren Mikrobefehlen (oder Mikrooperationen).

Als Benchmark-Programm wird ein einfaches Programm bezeichnet, das auf einer Reihe von unterschiedlichen Mikroprozessoren abgearbeitet werden kann, und sich dadurch ihre Leistungsfähigkeit vergleichen und auswerten läßt.

Monolithischer Mikroprozessor
Ein-Chip-Mikrocomputer, der Speicher, Rechenwerk, Leitwerk und Ein/Ausgabe enthält.

Mikroprogrammierung
Benutzer arbeitet auf dem fundamentalen Register-Transfer-Niveau

Mikrobefehls-Niveau
Benutzer definiert seinen Befehlsvorrat

Makrobefehls-Niveau
besteht aus mehreren Mikrobefehlen

Benchmark-Programme
Programme für die Bewertung der Leistungsfähigkeit unterschiedlicher Mikroprozessoren

Bild 4.1
Mikroprozessor-Fachsprache

Mikroprozessoren – Wann verwendet man sie und wann nicht?
(Bild 4.2)

Bild 4.2 zeigt Anforderungen, für die im besonderen der Einsatz eines Mikroprozessors passend erscheint: Werden gegenwärtig mehr als 30 TTL-Bausteine benötigt, sollte ein Mikroprozessor verwendet werden. Außerdem behauptet sich ein Mikroprozessor immer dann, wenn Programmierbarkeit für die Anpassung an vorhersehbare Systemumstellungen benötigt wird. Im Falle komplexer logischer/arithmetischer Funktionen fällt die Entscheidung für den Mikroprozessor, da es wesentlich einfacher ist, eine Reihe arithmetischer oder logischer Funktionen zu programmieren, als für diese eine fest verdrahtete Logik zu entwickeln.

Einige Punkte, in denen ein Mikroprozessor nicht verwendet werden sollte, zeigt ebenfalls Bild 4.2. Für hohe Arbeitsgeschwindigkeiten bietet der heutige Markt keine geeigneten Mikroprozessoren. So haben festverdrahtete ECL- oder Schottky-Steuerungen eine Ausführungszeit von 20 bis 150 ns und sind damit um ein bis zwei Größenordnungen schneller als die meisten erhältlichen Mikroprozessoren. Eine zweite Grenze für den Einsatz von Mikroprozessoren, speziell für monolithische Einheiten, sind große Wortlängen. Die maximale Wortlänge scheint heutzutage bei 32 Bit zu liegen. Ein drittes, für Mikroprozessoren wenig geeignetes Gebiet, sind Anwendungen bei denen extrem große Speicherkapazitäten benötigt werden, etwa ab 64 k Bytes und größer.

Schlüssel für das Verständnis von Mikroprozessoren
(Bild 4.3)

Es gibt zwei Hauptfaktoren für einen effektiven Mikroprozessorentwurf, die Architektur und der Befehlsvorrat. Die Architektur des Mikroprozessors befaßt sich mit der Definition der Funktionsblöcke, aus denen der Prozessor aufgebaut ist und wie diese Blöcke zusammenarbeiten.

Der zweite Faktor, der Befehlsvorrat, verdeutlicht, wie diese Funktionsblöcke zusammenwirken, da er das vom Hersteller beabsichtigte Einsatzfeld aufzeigt. Befehle sagen der Maschine, was zu tun ist. Werden Befehle in einer sinnvollen Folge aneinandergereiht, um spezifische Operationen auszuführen, spricht man von einem Programm. Je mehr Befehle ein Prozessor ausführen kann, desto vielseitiger ist er gewöhnlich.

Um das Verständnis der Architektur und des Befehlsvorrates zu vertiefen, betrachten wir, wie ein Mikroprozessor in einem System arbeitet. Grundsätzlich liegen zwei allgemeine Betriebsarten in einem System vor: der "Abfragemodus" und der "Unterbrechungsmodus".

"Abfragemodus" (polled operation)
(Bild 4.4)

Diese Betriebsart ist in Bild 4.4 dargestellt. Sie wird angewandt bei relativ langsam ablaufenden Vorgängen oder bei solchen, die auf ihre Bearbeitung warten können. Der Mikroprozessor fragt dabei einfach das Netzwerk (meist zyklisch) ab, bis er ein Gerät findet, das bedient werden muß. Findet er eines, wird das entsprechende Programm abgearbeitet und der Prozessor kehrt danach in den Abfragezyklus zurück.

Mikroprozessoreinsatz?

Ja, wenn:

- Mehr als 30 IC's benötigt werden
- Programmierbarkeit verlangt eird
- Komplexe Arithmetik oder Logik nötig ist

Nein, wenn:

- Hohe Arbeitsgeschwindigkeit: festverdrahtete ECL- oder Schottky-Steuerungen sind schneller
- Wortlänge größer als 32 Bit ist
- Großer Speicherbedarf: Anzahl der Adreßleitungen ist begrenzt

Bild 4.2
Mikroprozessoren: JA (+) oder NEIN (–)

Hauptfaktoren

- Architektur
- Befehlsvorrat

Bild 4.3
Schlüssel für das Verständnis von Mikroprozessoren

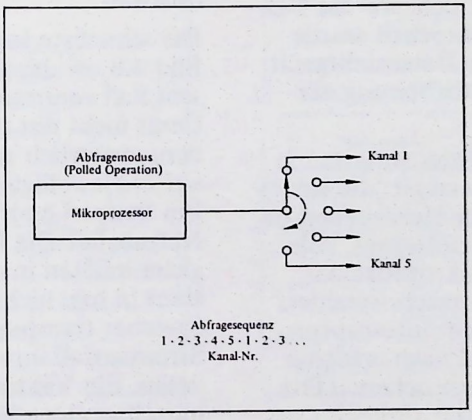


Bild 4.4
Abfrage-Modus

Unterbrechungs-Modus (Interrupt-Operation)
(Bild 4.5)

Die zweite Betriebsart eines Mikroprozessors ist der Unterbrechungsmodus. In diesem Fall wird die Zentraleinheit (CPU) durch ein Signal unterbrochen, um etwas spezielles zu tun, bevor sie mit dem nächsten Befehl (der unterbrochenen Sequenz) fortfährt. Typische Anwendungen für den Unterbrechungsmodus sind Echtzeit-Prozesse wie z.B. im Falle eines Datensichtgerätes (Video-Display-Terminal) entsprechend Bild 4.5. Werden Daten über die Tastatur zur exakt gleichen Zeit eingegeben, zu der der Mikroprozessor Daten im Speicher des Datensichtgerätes erneuern muß, damit dies seine Operationen fortführen kann, so unterbricht das Datensichtgerät einfach das Tastatur-Programm. Der Mikroprozessor merkt sich, wo das Tastatur-Programm unterbrochen wurde und bedient zuerst das Datensichtgerät. Danach kehrt er zur Bearbeitung der Tastatur zurück.

Es werden heute drei Interruptarten in Mikroprozessoren verwendet: die einzelne Interruptleitung, der Mehrebenen-Interrupt und der Vektorinterrupt. Alle drei veranlassen den Mikroprozessor, den Inhalt des Programmschrittzählers sicherzustellen, zu einem Interruptprogramm zu springen und nach erfolgter Abarbeitung zum unterbrochenen Programm zurückzukehren.

Einzelne Interruptleitung
(Bild 4.6)

In diesem Fall (Bild 4.6) besitzt der Mikroprozessor nur eine Interruptleitung und es müssen alle Geräte, die eine Unterbrechung auslösen können, über ODER-Gatter an diese eine Interruptleitung geschaltet sein. Jeder Eingang des ODER-Gatters kommt z.B. von einer Ein/Ausgabe-Einheit. Will eine dieser Einheiten bedient werden, sendet sie ein Signal aus, wodurch ein Interruptsignal erzeugt wird, das den Mikroprozessor veranlaßt, die Ein/Ausgabe-Einheit zu bedienen. Der Mikroprozessor muß hier jedoch, nachdem er unterbrochen wurde, alle Geräte abfragen um zu ermitteln, welches Gerät bedient werden muß.

Mehrebenen-Interrupt
(Bild 4.7)

Im Falle mehrerer Interruptleitungen (-ebenen) erfolgt die Identifikation des Gerätes, das eine Bedienung wünscht, unmittelbar, da der Mikroprozessor weiß, welche Leitung die Unterbrechung verursacht hat; er braucht also nicht alle Geräte abzufragen (siehe Bild 4.7).

Vektorinterrupt
(Bild 4.8)

Die schnellste Interruptart, skizziert in Bild 4.8, ist der Vektorinterrupt. In diesem Fall verursacht das unterbrechende Gerät nicht nur ein Interruptsignal, sondern zusätzlich einen direkten Sprung auf das jeweilige Interrupt-Programm. Ein Beispiel hierfür ist ein auftretender Netzausfall Die Inhalte kritischer Register müßten in diesem Fall schnellstens in sog. nichtflüchtige (nonvolatile) Speicher transportiert werden, die bei Stromausfall ihre Information nicht verlieren. Ein Vektorinterrupt würde unmittelbar einen Sprung zu einem entsprechenden Programm bewirken.

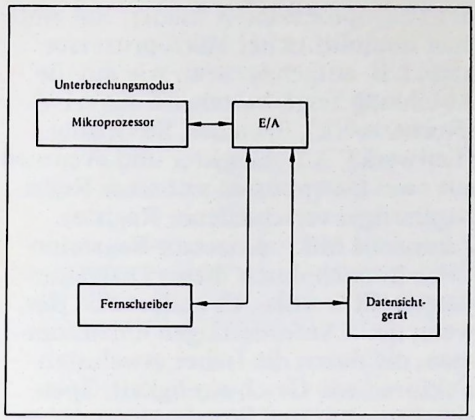


Bild 4.5
Unterbrechungs-Modus

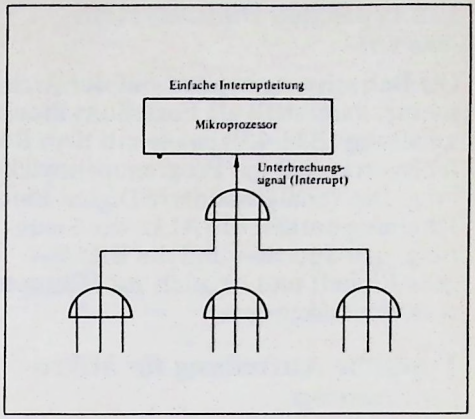


Bild 4.6
Einfache Interruptleitung

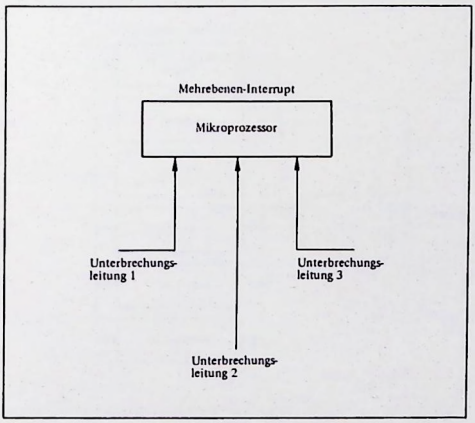


Bild 4.7
Mehrebenen-Interrupt

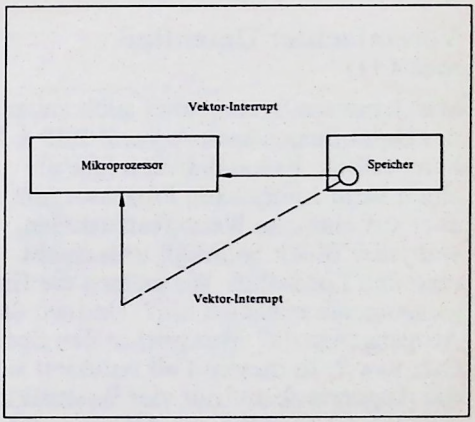


Bild 4.8
Vektorinterrupt

Ein typisches Digitalsystem
(Bild 4.9)

Die Betrachtung beginnt mit der Architektur, dargestellt als Funktions-Blockschaltung (Bild 4.9) sowie mit dem Befehlsvorrat und der Programmentwicklung. Die verallgemeinerte Digital-Blockschaltung enthält die ALU, die Steuerung, den Speicher und die Ein/Ausgabe-Einheit und ist auch auf Mikroprozessoren anwendbar.

Typische Aufteilung für Mikroprozessoren
(Bild 4.10)

Verschiedene Architekturen teilen das Digitalsystem in verschiedener Art und Weise auf. Bild 4.10 zeigt eine gängige Art der Aufteilung eines Mikroprozessors.

Vereinfachter Datenfluß
(Bild 4.11)

Man kann ein System aber auch anhand des Datenflusses entsprechend Bild 4.11 beschreiben. Es handelt sich hier um einen recht komplexen Prozessor; um aber auf einfache Weise festzustellen, was jeder Block benötigt, untersucht man den Datenfluß. Wo müssen die Eingangssignale zunächst hin? Und wo die Ausgangssignale? Was geht in den Speicher usw.? In diesem Fall reduziert sich das Augenmerk auf nur vier Busstrukturen, auf die Eingabe, die Ausgabe, den Speicher und die Befehlsinformation. Bild 4.11 repräsentiert keinen universell gültigen Datenfluß; es soll nur den zweiten Weg, die Betrachtung der Architektur über den Datenfluß, veranschaulichen.

Ein typischer monolithischer Mikroprozessor
(Bild 4.12)

Bild 4.12 zeigt die Funktionsblock-Anordnung eines typischen monolithischen Mikroprozessors und gibt damit in einer mehr spezifischen Weise die Architektur der Mikroprozessoren wieder. Ein einfacher monolithischer Mikroprozessor kann z.B. aufgebaut sein, wie ihn die Abbildung zeigt, bestehend aus ALU (Rechenwerk), Speicher, Steuerung (Leitwerk), Adreßregister und eventuell aus zwei Datenbussen mit einer Reihe zugehöriger verschiedener Register. Zahlreiche Mikroprozessor-Bausteine-Sätze können durch dieses Diagramm dargestellt werden. Es ändert sich nur, wenn neue Anforderungen hinzukommen, die durch die früher erwähnten Faktoren wie Geschwindigkeit, Speicherkapazität und Wortlänge eingeschränkt werden.

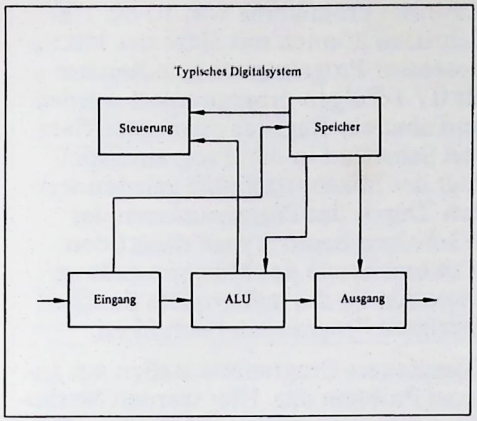


Bild 4.9
Typisches Digital-System

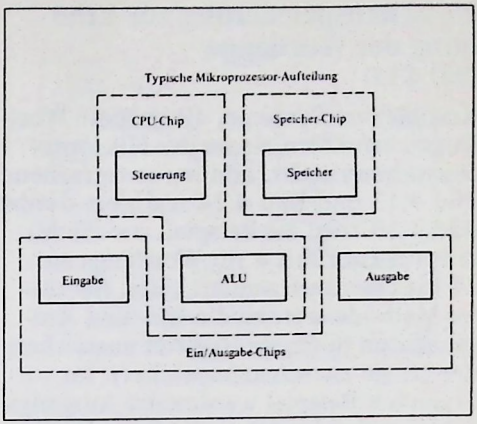


Bild 4.10
Typische Mikroprozessor-Aufteilung

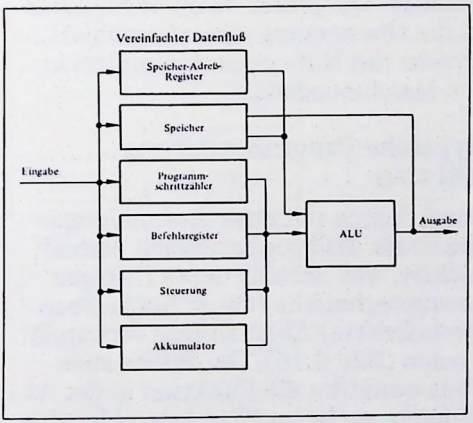


Bild 4.11
Vereinfachter Datenfluß

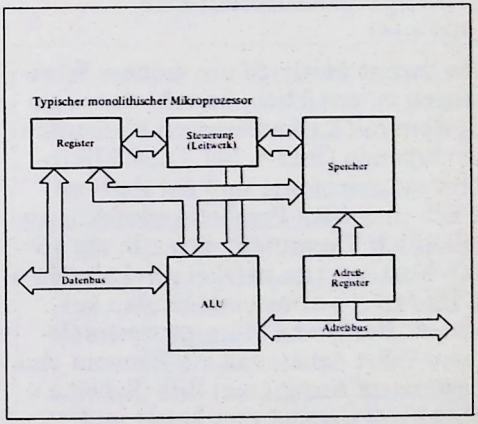


Bild 4.12
Typischer monolithischer Mikroprozessor

Zwischenspeicherung zur Erhöhung der Wortlänge
(Bild 4.13)

Komplexere Systeme, die größere Wortlängen erfordern als sie der Mikroprozessor bereitstellt, können entsprechend Bild 4.13 und Bild 4.14 realisiert werden. Bild 4.13 zeigt als Beispiel, wie ein Mikroprozessor mit 8 Bit Wortlänge auf 24 Bit erweitert werden kann. Bei dieser Methode werden die Ein- und Ausgabedaten in einem Register ausreichender Länge zwischengespeichert. Im vorliegenden Beispiel werden die Ausgangsdaten nacheinander in die 8-Bit-Register gebracht. Da hier mehrere Zyklen, in diesem Fall drei, zur Abwicklung einer bestimmten Operation notwendig sind, verlangsamt sich das gesamte System.

Mikroprozessorelemente
(Bild 4.14)

Die zweite Methode um größere Wortlängen zu erreichen, besteht darin, ein System mit kaskadierbaren Elementen aufzubauen (Bild 4.14). Diese Mikroprozessorelemente sind zur Zeit mit 2-Bit- und 4-Bit-Parallel-Architekturen erhältlich. Damit läßt sich z.B. ein 16-Bit-Wort aus vier parallel geschalteten 4-Bit-Mikroprozessorelementen aufbauen. Der Name Mikroprozessorelement rührt daher, daß ein Element eine bestimmte Anzahl von Bits (Scheibe ≙ Slice) voll parallel verarbeitet und gleichzeitig ein Übertrags-Bit des vorhergehenden Elementes mitverarbeiten kann. Die verbleibenden Bits werden von anderen Elementen verarbeitet.

Erstellung der Mikroprozessor-Software
(Bild 4.15)

Mit Bild 4.15 beginnt die Besprechung des verbleibenden Hauptfaktors der Projekt-Definition, dem Befehlsvorrat bzw. der Software. Wie die Abbildung zeigt, gibt es drei Wege Mikroprozessor-Software zu erstellen.

Einfache Programme von 10 bis 100 Schritten können mit Hilfe des Mikroprozessor-Programmier-Handbuches als 0 / 1-Folgen programmiert werden und über ein Register oder einen Satz von Schaltern in den Programmspeicher des Mikroprozessors geladen werden. Durch das Programmieren der 0/1-Folgen benutzt man direkt den Maschinencode des Mikroprozessors. Diese Art ist die mühevollste der drei gezeigten Programmiertechniken.

Komplexere Programme stellen ein anderes Problem dar. Hier werden Methoden bevorzugt, die nicht bitweises Programmieren der Befehle erfordern (0/1), sondern mit Problem- oder Prozedur-Sprachelementen arbeiten. Dies verlangt automatische Wege für die Umsetzung der Programmiersprache in den Maschinencode. Die gebräuchlichste Methode ist die Übersetzung einer Assemblersprache mit Hilfe eines Assemblers in den Maschinencode.

Typische Programmierung
(Bild 4.16)

Das Arbeiten mit einer Assemblersprache ist für das Programmieren deshalb leichter, weil anstelle der 0/1-Folgen mnemotechnische (durch Buchstaben symbolisierte) Abkürzungen verwendet werden (Bild 4.16). Der Buchstaben-Code bestimmt die Funktion in der Assemblersprache und das Assembler-Programm übersetzt dies in den Maschinencode.

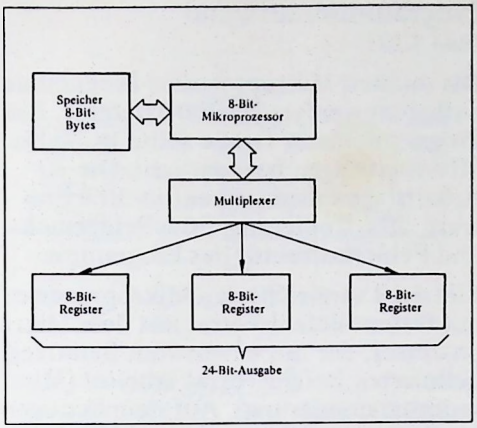


Bild 4.13
Zwischenspeicherung zur Erhöhung der Wortlänge

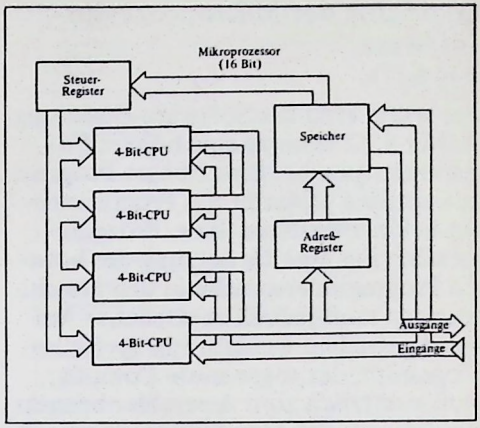


Bild 4.14
Mikroprozessor, aufgebaut mit Mikroprozessorelementen

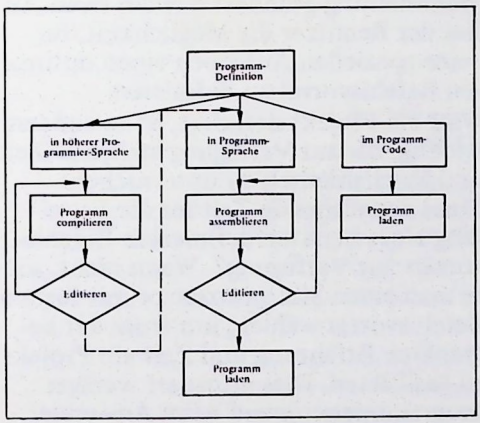


Bild 4.15
Erstellung der Mikroprozessor-Software

Typische Programmierung

Assemblersprache		Maschinencode	
Mnemonic	Funktion	Befehl	Adresse
LDA	Lade Register A	0001	1001
HLT	Halt (Anhalten der CPU)	0000	0000

Bild 4.16
Typische Programmierung

Erstellung der Mikroprozessor-Software
(Bild 4.17)

Der letzte Pfad zur Software-Erstellung in Bild 4.17 benützt eine höhere Programmiersprache. Eine höhere Programmiersprache gestattet das Programmieren in einer Problem- bzw. Prozedursprache. Für eine Umsetzung der höheren Programmiersprache in den Maschinencode muß jedoch zusätzlicher Aufwand getrieben werden. Ein spezielles Programm, der sogenannte Compiler, muß zusätzlich zum Assembler benutzt werden, um den Maschinencode zu erhalten.

Zur Zeit ist nur eine begrenzte Anzahl von Mikroprozessoren erhältlich, die diese Programmierung erlauben.

Schritte der Programmierung
(Bild 4.18)

Da das Programmieren in Assemblersprache die gängigste Methode ist, zeigt Bild 4.18 einige Details, wie ein solches Programm erstellt werden kann. Zuerst wird ein Flußdiagramm der Aufgabe angefertigt, jeder Programmschritt und jede Manipulation wird so definiert, daß der Programmierer daraus leicht den Code für die Aufgabe schreiben kann. Dann wird ein Assembler für die Umsetzung in den Maschinencode des Mikroprozessors benutzt. Ist das Programm sehr komplex, wird es auf einem (größeren) Digitalrechner simuliert. Erfüllt es alle Anforderungen (fehlerfrei), wird es im nächsten Schritt geladen. Im anderen Fall muß es eine Phase der Fehlersuche (Debug/Debugging) und Programmänderung durchlaufen, und nochmals die Schleife durchlaufen (Bild 4.18).

Programmieraufwand
(Bild 4.19)

Die meisten Mikroprozessor-Programme enthalten weniger als 2000 Befehle. Ein Programm dieser Größe sollte in 20 bis 30 Arbeitstagen beendet sein. Die 30 Arbeitstage teilen sich auf in 30% Entwurf, 20% Codierung, 50% Fehlersuche und Fehlerkorrektur des Programms.

Bild 4.19 vergleicht den Mikroprozessor mit festem Befehlsvorrat mit dem Mikroprozessor, der mit einem vom Benutzer definierten Befehlsvorrat arbeitet (Mikroprogrammierung). Auf dem heutigen Markt finden sich vorwiegend Mikroprozessoren mit festem Befehlsvorrat. Der Benutzer hat keinen Zugang zu den Mikrobefehlen, sondern nur zu den Makrobefehlen, kurz Befehle genannt. Jedoch gibt es auch einige Mikroprozessoren, deren Befehlsvorrat durch Mikroprogrammierung geändert werden kann. So hat der Benutzer die Möglichkeit, für seine speziellen Aufgaben einen optimalen Befehlsvorrat zu definieren.
Wird ein Projekt definiert, ist es äußerst wichtig, die zur Verfügung stehende Zeit und Möglichkeiten zu untersuchen: Steht ausreichende Zeit für die Erstellung eines benutzerdefinierten Befehlsvorrats zur Verfügung? Wenn nicht, sollte man einen Mikroprozessor mit festem Befehlsvorrat wählen, um trotz der begrenzten Erfahrung und Zeit ein Projekt zu realisieren. Dies erfordert weniger Programmieraufwand beim Anwender, verlangt jedoch umfangreiche Softwareunterstützung durch den Hersteller.

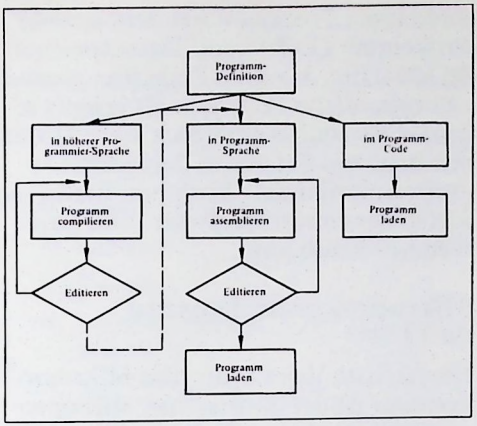


Bild 4.17
Erstellung der Mikroprozessor-Software

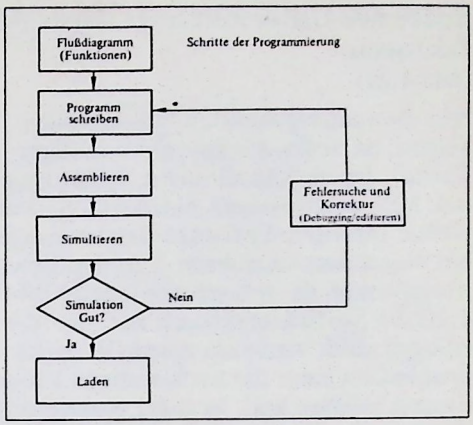


Bild 4.18
Schritte der Programmierung

- Mikrobefehlsniveau
  sehr großer Aufwand
- Makrobefehls-Niveau
  fester Befehlsvorrat
  wenig Aufwand

Bild 4.19
Programmier-Aufwand

Software-Unterstützung: Auswahlfaktoren
(Bild 4.20)

Wie die vorhergehenden Diskussionen zeigen, ist Software ein sehr wichtiger Faktor. Es ist deshalb nicht länger möglich Mikroprozessoren einzusetzen, ohne vorher intensive Software-Betrachtungen bei ihnen durchzuführen. Um den Bewertungsprozeß zu unterstützen, zeigt Bild 4.20 die Softwarefaktoren auf, die festzulegen sind, wenn ein spezielles Mikroprozessorsystem die notwendigen Forderungen erfüllen soll. In jeder Anwendung werden alle oder nur ein Teil der angegebenen Programme benötigt!

Typischer monolithischer Mikroprozessor
(Bild 4.21)

Zunächst sollen die zur Zeit erhältlichen Produkte sortiert werden. Das Marktangebot läßt sich in vier Gruppen einteilen, deren erste die in sich abgeschlossenen monolithischen Mikroprozessoren (Ein-Chip-Mikrocomputer) sind (Bild 4.21). Eingesetzt werden diese Geräte vorwiegend in programmierten Steuerwerken, Taschen- und Tischrechnern sowie in verschiedenen Anwendungen, bei denen der Programmieraufwand begrenzt ist und keine hohen Geschwindigkeitsanforderungen stellen. Typisch ist, daß diese Prozessoren mit der Maskenerstellung programmiert werden (ROM-Speicher) und einen begrenzten Datenspeicher (RAM) enthalten. Sie repräsentieren den einfachsten Weg einen Mikroprozessor einzusetzen, da der Hauptteil des einfachsten Entwicklungsaufwandes vom Hersteller übernommen worden ist.

Typische Mikroprozessor-Aufteilung
(Bild 4.22)

Die zweite Gruppe von Mikroprozessoren auf dem Markt ist der Baustein-Satz, wie ihn Bild 4.22 zeigt. Dies ist ein sehr vielseitiger Prozessor, bei dem sowohl Programm- (ROM) und Datenspeicher (RAM), Ein/Ausgabe-Einheiten als auch der eigentliche Prozessor als jeweils separate Bausteine erhältlich sind. Mit diesen Baustein-Sätzen ist der Interface-Aufwand minimal. Auch hier liefert der Hersteller starke Hardware- und Softwareunterstützung.

Mikroprozessorelemente
(Bild 4.23)

Die nächsthöhere Stufe von Mikroprozessoren ist der Aufbau mit Mikroprozessorelementen, skizziert in Bild 4.23. Der Hersteller liefert hierbei ein Mikroprozessorelement, welches voll kaskadierbar ist. Mikroprozessorelemente wurden entwickelt, da die erreichbare Komplexität eines Halbleiterkristalls begrenzt ist, speziell bei Technologien, die hohe Verarbeitungsgeschwindigkeiten ermöglichen. Typisch für diese Art ist der Minicomputer-ähnliche Aufwand, der zu treiben ist, da hier sowohl die Programmierung als auch die Schnittstellen (Interfaces) sehr komplexe Aufgaben darstellen. Allgemein gesprochen, ist die Unterstützung durch den Hersteller in dieser Gruppe nicht so vollkommen, da diese Prozessoren sehr vielseitige Entwürfe zulassen und es nahezu unmöglich ist, jeden möglichen Entwurf zu unterstützen.

- Assembler
- Editor
- Debugger
- Loader (= Lader)
- Simulator
- Compiler
- Hardware/Software-Entwicklungssystem

Bild 4.20
Software-Unterstützung: Auswahl

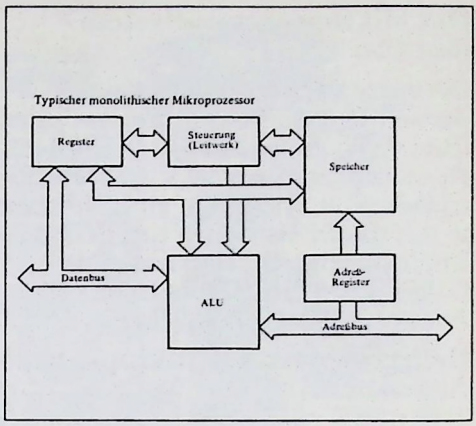


Bild 4.21
Typischer monolithischer Mikroprozessor

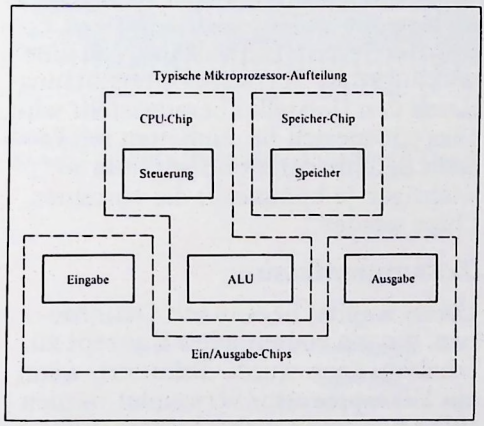


Bild 4.22
Typische Mikroprozessor-Aufteilung

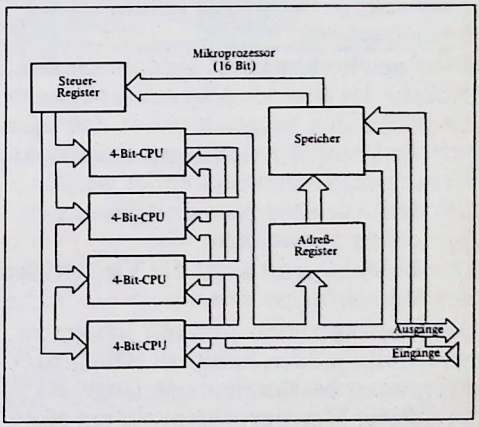


Bild 4.23
Mikroprozessor, aufgebaut mit Mikroprozessorelementen

Das Mikroprozessor-System
(Bild 4.24)

Die vierte zu betrachtende Gruppe ist das vollständige Mikroprozessor-System (Bild 4.24). In diesem Fall liefert der Hersteller das komplett aufgebaute betriebsbereite System auf einer Schaltkarte mit starker Hardware- und Software-Unterstützung. Oft sind Assembler-, Editor- und DEBUG-Programme in Form von ROMs erhältlich.

Richtlinien für die Mikroprozessor-Auswahl
(Bild 4.25)

Bild 4.25 zählt alle Faktoren nochmals auf, die bei der Wahl eines Mikroprozessors betrachtet werden sollen.

Der erste Faktor ist die **Wortlänge**. Sie steht in enger Beziehung zum verlangten Datenumfang bzw. zur verlangten Genauigkeit.

Die **Speicherkapazität** wird durch die Anzahl der Bits im Adressbus begrenzt. Es wird dabei ausgeschlossen, daß irgendwelche komplizierten Mehr-Zyklen-Adressierungstechniken verwendet werden, um den Adressbereich und damit den Speicher zu erweitern.

Der **Befehlsvorrat** zeigt die Vielseitigkeit des Mikroprozessors an.

Ausführungszeiten sind nur schwer zu ermitteln; jedoch kann ein Benchmark-Programm bestimmen, wie lange ein ausgewählter Mikroprozessor für ein spezielles Beispiel jeweils braucht. Somit kann die richtige Einheit für den Anwendungsfall ausgewählt werden.

Die Anzahl der Register ist ein Maß für die Vielseitigkeit der Programmierung. Diese ist um so größer, je mehr Register für Adressierungszwecke (z.B. Index- oder Adressregister) verwendet werden können.

Ein anderer Faktor für die Vielseitigkeit bei der Programmierung ist meist die Anzahl der **Adressierarten**. Je mehr Arten bei einem Mikroprozessor möglich sind, desto größer ist die Vielseitigkeit des Systems.

Der **direkte Speicherzugriff (DMA)** steht in enger Beziehung zur Daten- bzw. Programm-Übertragungsrate. DMA ist speziell bei Anwendungen in denen große Datenmengen transportiert werden, wünschenswert. Für schubweise Verarbeitung von Daten, wie sie bei Datensichtgeräten angewendet wird, ist DMA unerläßlich.

Die Größe des **Stack-Speichers** ist ein Maß für Ausführungszeiten und einfache Programmierung. Ein großer und vielseitiger Stack-Speicher erlaubt nahezu unbegrenzt viele Unterprogramm-Verschachtelungen und bietet mehr Freiheiten in der Programmierung.

Die **Unterstützung des Herstellers**, der Aufwand an **zusätzlichen Bausteinen**, die **Gehäusegröße**, die **elektrischen Betriebsdaten**, **Zweithersteller** und **Preisvergleiche** bedürfen keiner weiteren Erklärung. Es ist sehr wichtig für den Erfolg des System-Entwicklers, daß eine wirkungsvolle Software-Unterstützung durch den Hersteller bereitgestellt wird. Dies gilt speziell für Einheiten mit festem Befehlsvorrat und wird um so wichtiger, je komplexer die einzelnen Chips werden.

Zusammenfassung

Dieses Kapitel begann mit Definitionen, um ein einheitliches Konzept zu erhalten; dann wurde diskutiert, wann ein Mikroprozessor verwendet werden sollte und wann nicht. System- und Hardwareaspekte der Projektdefinition wurden behandelt sowie die Software-(Programm-) Definition. Das Kapitel endete mit einer Diskussion der Faktoren, die bei der Mikroprozessorauswahl eine Rolle spielen.

Mit diesem Kapitel sollte man in der Lage sein, den Rahmen eines Mikroprozessor-Projekts festzulegen, mit dem Hersteller über die Anforderungen zu sprechen und den richtigen Mikroprozessor für die spezifische Anwendung auszuwählen.

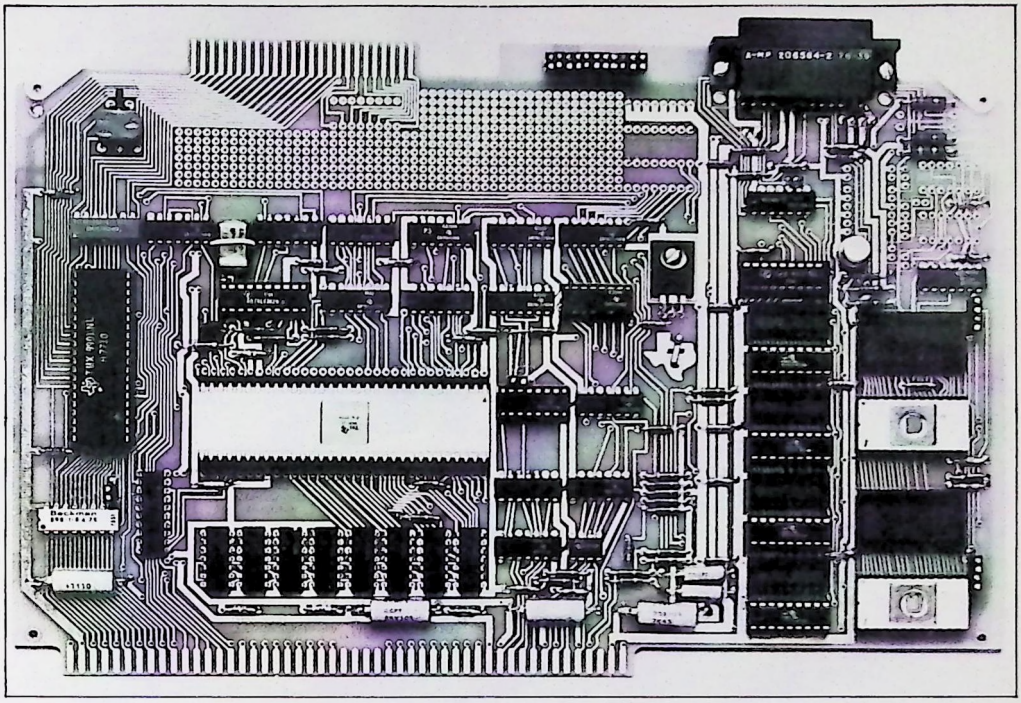


Bild 4.24
Beispiel für ein Mikroprozessorsystem: Das TM 990/100M Mikrosystem

Wortlänge	Größe des Stacks
Speicher-kapazität	Hersteller-Unterstützung
Befehlsvorrat	Zusätzlich notwendige Bausteine
Ausführungszeit	Gehäusegröße
Anzahl der CPU-Register	Elektrische Betriebsarten
Adressierungsarten	Zweithersteller
Intrrupts	Preis
DMA-Möglichkeit	

Bild 4.25
Richtlinien für die Mikroprozessorauswahl

KAPITEL 5

SOFTWARE FÜR MIKROPROZESSOREN

In diesem Kapitel werden Ausdrücke definiert und der grundlegende Software-Systementwurf behandelt mit dem Ziel, dem Leser diejenigen Kenntnisse zu vermitteln, die er benötigt, um ein Softwarepaket für eine spezielle Anwendung zu entwickeln.

Software für Mikroprozessoren

Dieses Kapitel untersucht den Befehlsvorrat eines Mikroprozessors und was es bedeutet, ein Programm zu schreiben. Die Programme für einen Rechner werden Software genannt. Die Hauptaufgabe dieses Kapitels besteht darin, das Vokabular der Mikroprozessor-Software zu erklären und zu zeigen, wie diese erzeugt werden kann.

Die zwei grundsätzlichen Software-Komponenten sind die Anwendungs- und die Systemprogramme. Anwender-Programme sind solche Programme, die eine vom Anwender erwünschte Funktion ausführen. Dies sind Programme, die von Benützer zu Benützer verschieden sind. Systemprogramme werden deshalb entwickelt, um Rechner und damit auch Mikroprozessor-Systeme besser an die Bedürfnisse eines Anwenders anzupassen. Einige Beispiele für Systemprogramme sind Loader (Lader), Compiler und Betriebssysteme.

Sprachniveaus
(Bild 5.1)

Um einen Mikrocomputer zu programmieren, muß eine Liste von codierten Befehlen erstellt werden. Diese Befehle müssen in einer bestimmten Form bzw. in einer Sprache vorliegen, die der Prozessor versteht. Eine Computersprache ist eine Sprache, anhand der einem Computer Befehle übermittelt werden, damit dieser die gewünschten Funktionen ausführt. Im Laufe der Zeit haben sich mehrere Niveaus von Prozessorsprachen mit dem Ziel herausgebildet, diese für den Rechner verständlich zu machen. Sie sind in Bild 5.1 angegeben.

Das niedrigste Sprachniveau, die Maschinensprache, wird direkt von der Zentraleinheit des Rechners übersetzt; sie besteht aus Dualzahlen, also aus einer Folge von Nullen und Einsen. Die Null- und Eins-Kombinationen spezifizieren die unterschiedlichen Computerfunktionen. Deshalb ist es natürlich möglich, einen Mikroprozessor im Dualcode zu programmieren.

Die mit Assembler bezeichneten Sprachen wurden entwickelt, um durch die Verwendung von Buchstaben die Anweisungen der Maschinensprache symbolisch darstellen zu können. In der Assemblersprache ist jedem Operationscode ein Buchstaben- bzw. Mnemonic-Code zugeteilt, wie z.B.: ADD, STORE, JUMP oder LDA. Um nun die Assemblersprache in die Maschinensprache zu übersetzen, wird ein Programm benötigt. Dieses Programm wird als Assembler bezeichnet.

Die höheren Programmiersprachen können in zwei weitgefaßte Kategorien eingeteilt werden: in verfahrensorientierte Programmiersprachen wie FORTRAN, COBOL, PL/1 und PL/M sowie in problemorientierte Programmiersprachen wie GPSS, SIMSCRIPT, LISP und SNOBOL.

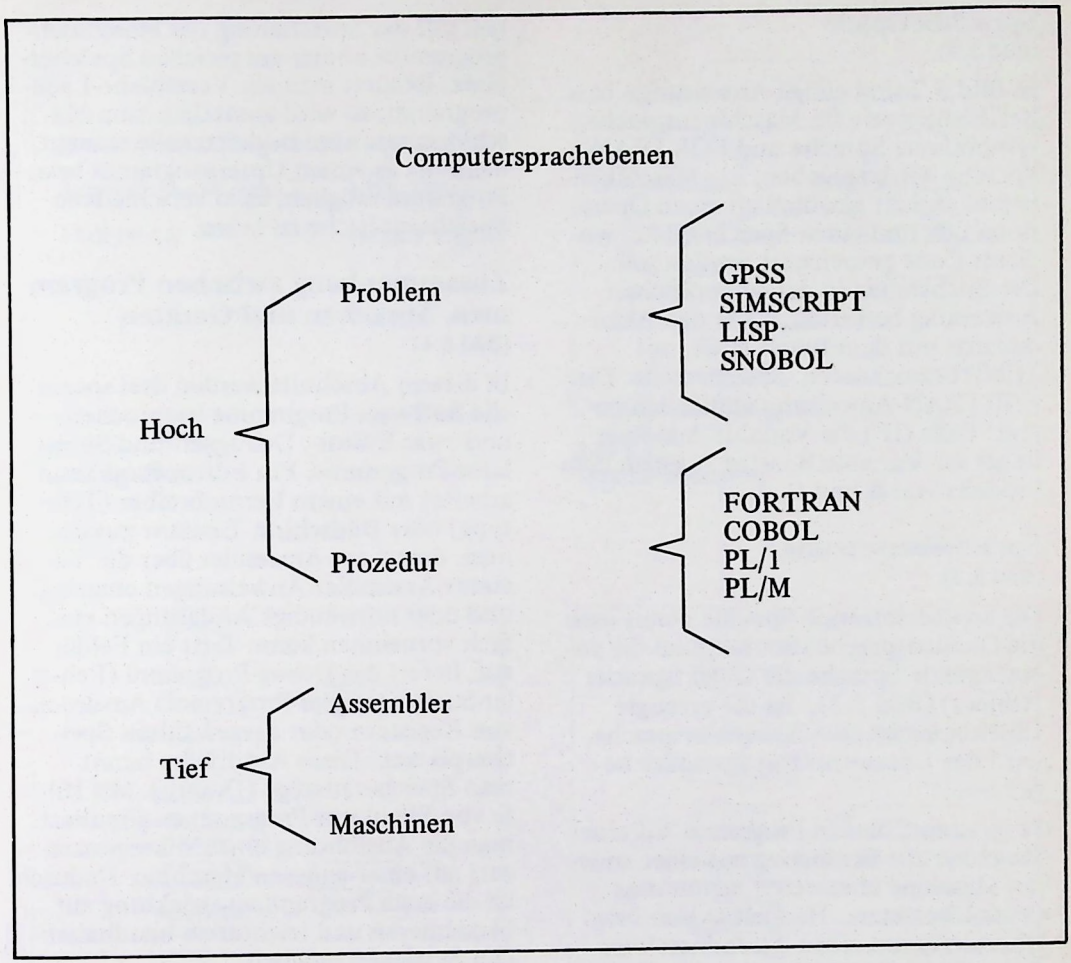


Bild 5.1
Computersprachebenen

Sprachbeispiele
(Bild 5.2)

In Bild 5.2 sind einige Anweisungs- bzw. Befehlsbeispiele für Maschinensprache, symbolische Sprache und FORTRAN-Sprache wiedergegeben. Ein Maschinenbefehl enthält gewöhnlich einen Operationscode und einen Speicherplatz, wo dieser Code gespeichert werden soll. Die Buchstaben in der symbolischen Anweisung bedeuten: Lade den Akkumulator mit dem Inhalt einer, mit TEMP bezeichneten, Speicherzelle. Die FORTRAN-Anweisung schließlich besagt: Falls (IF) die Variable A kleiner ist als die Variable B, setze A gleich dem Produkt von B und C.

Sprachübersetzung
(Bild 5.3)

Die zu übersetzende Sprache nennt man die Quellensprache (Source) und die zu erzeugende Sprache die Objektsprache (Object) (Bild 5.3). Ist die erzeugte Objektsprache eine Assemblersprache, wird der Übersetzer mit Compiler bezeichnet.

Programme, die ein Programm auf einer Maschine zur Benützung auf einer anderen Maschine übersetzen, nennt man Cross-Übersetzer. Handelt es sich beim Übersetzer um einen Assembler, wird dieser Cross-Assembler genannt. Ein als Cross-Assembler geschriebenes Programm arbeitet meist auf einer größeren Anlage, aber der erzeugte Objektcode ist ein Maschinencode für kleinere Systeme.

Der von Assemblern und Compilern erzeugte (generierte) Maschinencode wird gewöhnlich in Halbleiterspeicher (RAM/ROM) geladen. Hierzu benötigt man ein sog. Lade-Programm (LOADER). Wir besprechen in diesem Kapitel zwei Lader-Typen, den absoluten Lader (absolut LOADER) und das Ladeprogramm für verschiebbare Programme (relocatable LOADER). Absolute Lader beginnen mit der Speicherung des Maschinenprogramms immer am gleichen Speicherplatz. Benützt man ein Verschiebe-Ladeprogramm, so wird zusätzlich zum Maschinencode eine Begleittabelle erzeugt, damit ist es einem Unterprogramm bzw. Programm möglich, es in verschiedene Speicherbereiche zu laden.

Zusammenhang zwischen Programmen, Sprachen und Geräten
(Bild 5.4)

In diesem Abschnitt werden drei spezielle Software-Programme besprochen, und zwar Editor-, Debugger- und Simulator-Programme. Ein Editor-Programm arbeitet mit einem Fernschreiber (Teletype) oder Bildschirm-Tastatur zusammen, damit der Anwender über die Tastatur Assembler-Anweisungen eingeben und oder notwendige Änderungen einfach vornehmen kann. Tritt ein Fehler auf, liefert das Debug-Programm (Fehler-Such/Korrigier-Programm) Ausdrucke von Registern oder ausgewählten Speicherplätzen. Diese Ausdrucke nennt man Speicherauszüge (Dumps). Mit Hilfe von Simulator-Programmen simuliert man die Ausführung eines Mikroprozessors auf einer anderen Maschine. Dadurch ist die erste Programmentwicklung auf bequemeren und leichter zu handhabenden Systemen möglich.

Bild 5.4 zeigt, wie Sprachen und andere Software-Komponenten zueinander in Beziehung stehen. In Bild 5.4a ist in vereinfachter Form dargestellt, wie ein in einer höheren Sprache geschriebenes Programm in einen Halbleiterspeicher, z.B. in das RAM eines Mikroprozessors, geladen wird. Aus Bild 5.4b gehen die Änderungen hervor, die notwendig sind, wenn die Eingabe des Programmes in den Prozessor in Assemblersprache erfolgt. Hierbei findet der EDIT-Vorgang im Assembler-Niveau statt.

Maschinensprache	11001011
Symbolische Sprache	LDA TEMP
FORTRAN	IF (A.LT.B) A = B * C

Bild 5.2
Sprachbeispiele

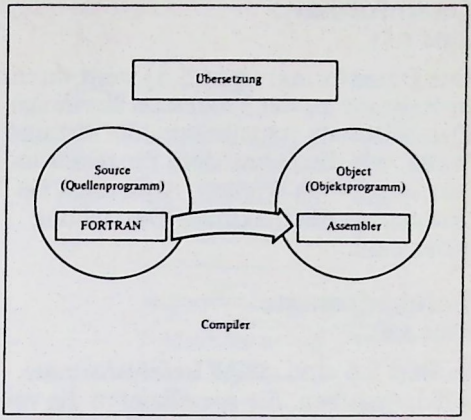
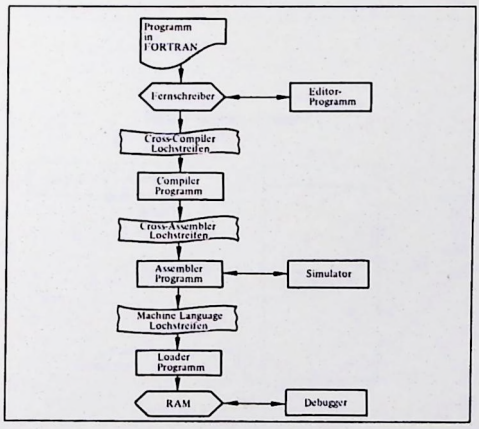


Bild 5.3
Sprachübersetzung



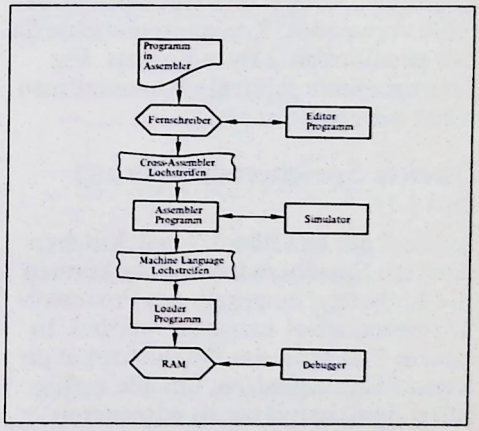


Bild 5.4
Zusammenhang zwischen Programmen, Sprachen und Geräten

Datenformat
(Bild 5.5)

Das Datenformat (Bild 5.5) steht direkt in Relation zu der Prozessor-Wortlänge. Datenformate spezifizieren die Art und Weise, wie die Daten dem Prozessor intern angeboten werden. Außerdem begrenzen sie die maximale Darstellung einer Zahl.

Befehlsformate
(Bild 5.6)

In Bild 5.6 sind einige Befehlsformate wiedergegeben. Sie spezifizieren die verschiedenen Wege zur Darstellung unterschiedlicher Prozessor-Befehle. Häufig beanspruchen Befehle mehr als ein Wort, besonders bei kleinen Wortlängen. Normalerweise werden 1-Byte-Formate für Register-zu-Register-, Ein/Ausgabe-, Logische-, Rotier- und Rücksprung-Befehle verwendet. Konstantenbefehle haben gewöhnlich 2-Byte-Format. Für Sprungbefehle schließlich benützt man meist 3-Byte-Formate.

Direkte Speicheradressierung
(Bild 5.7)

Anhand der aus Bild 5.7 ersichtlichen direkten Speicheradressierung können alle Einheiten innerhalb des Prozessor-Adressbereiches adressiert werden. In diesem Fall muß das Befehlsformat genügend Bits enthalten, um alle verfügbaren Speicherplätze zu adressieren.

Immediate-Adressierung
(Bild 5.8)

Bei der Immediate-Adressierung ist der Operand im Befehl spezifiziert. Beispielsweise wird durch den Befehl in Bild 5.8 der Wert von "00100" (Basis 16) in den Akkumulator gegeben. Sie ist die schnellste Adressiermethode, da keine weiteren Speicherzugriffe, außer dem Anfangs-Befehlsabruf, erforderlich ist. Allerdings wird die Größe des Operanden durch das Befehlsformat bestimmt.

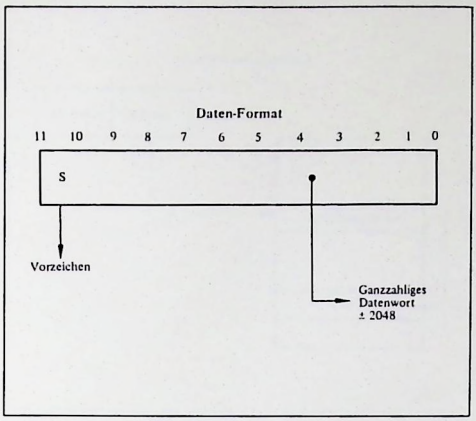


Bild 5.5
Datenformat

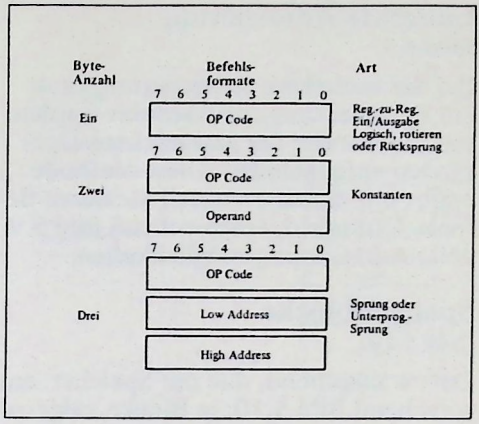


Bild 5.6
Befehlsformate

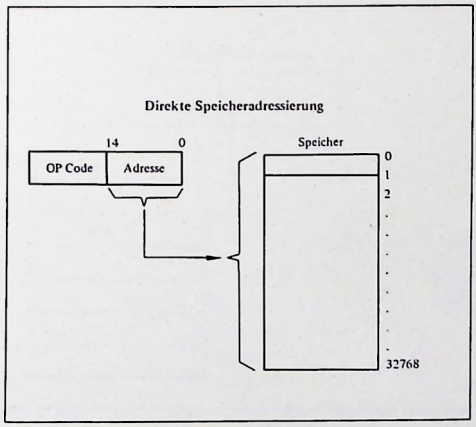


Bild 5.7
Direkte Speicheradressierung.

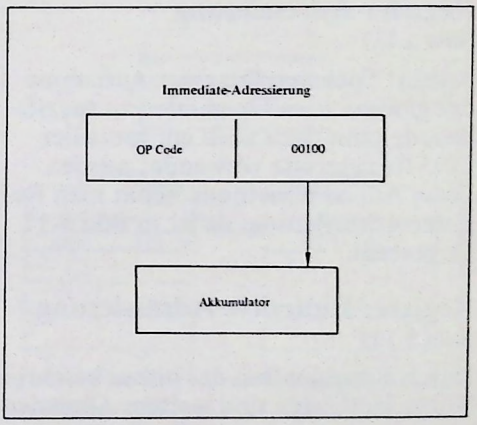


Bild 5.8
Immediate-Adressierung

Indirekte Adressierung
(Bild 5.9)

Bei der indirekten Adressierung kann ein Speicherwort dazu benutzt werden, um die Adresse der gewünschten Operanden aufzunehmen. Diese Methode ergibt mit einem eventuell kleineren Befehls-Adressfeld, entsprechend Bild 5.9, volle Adressierungsmöglichkeiten.

Speicherblöcke
(Bild 5.10)

Davon ausgehend, daß der Speicher, entsprechend Bild 5.10, in Blöcke aufgeteilt wird, zeigt das indirekte Adressfeld in einem Befehl auf einen Speicherplatz innerhalb der laufenden Seite Page 0. Der Inhalt dieser einzelnen Seite umfaßt den aktuellen Operanden-Ort.

Register-Adressierung
(Bild 5.11)

Anstatt Speicherplätze zur Aufnahme der gewünschten Operanden zu spezifizieren, kann dazu auch ein spezieller CPU-Registersatz verwendet werden. Diese Adressiermethode nennt man Register-Adressierung; sie ist in Bild 5.11 dargestellt.

Register-Indirekte-Adressierung
(Bild 5.12)

Durch Kombination der bisher beschriebenen Methoden sind weitere Adressierarten möglich, wie z.B. die Register-Indirekte-Adressierung. In diesem Fall schließt beispielsweise das Befehlsformat ein Feld zur Spezifizierung eines Registers ein, dessen Inhalt eine indirekte Adresse darstellt, wie in Bild 5.12 gezeigt.

Die verschiedenen Mikroprozessor-Befehlsarten werden in vier Klassen eingeteilt: arithmetische und logische Befehle, Datentransportbefehle, bedingte und unbedingte Sprungbefehle und spezielle Befehle.

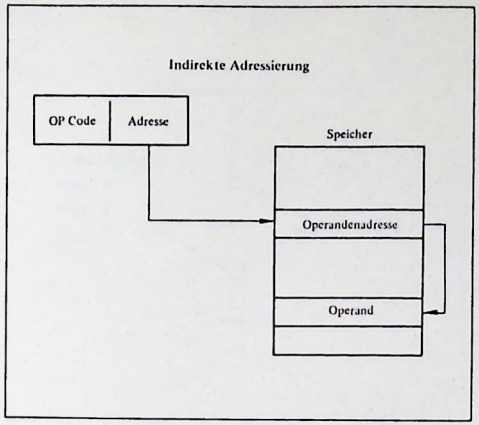


Bild 5.9
Indirekte Adressierung

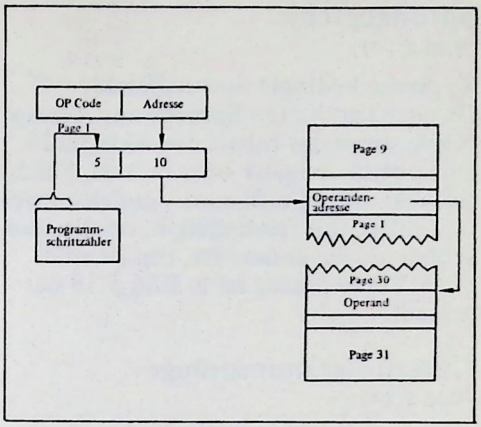


Bild 5.10
Speicherblöcke

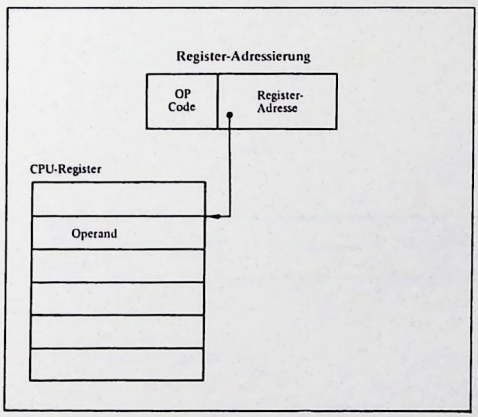


Bild 5.11
Register Adressierung

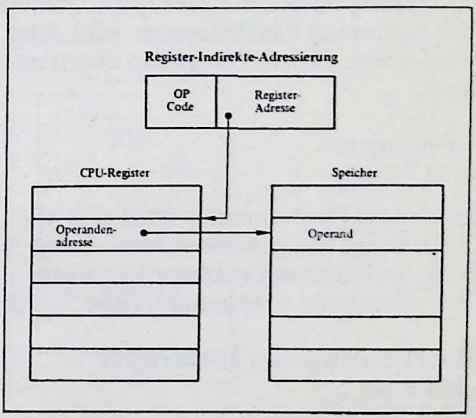


Bild 5.12
Register-Indirekte Adressierung

Sprungbefehle
(Bild 5.13)

Typische bedingte Sprungbefehle (Branch) enthalten Sprünge zur Adresse XXX, wenn der Inhalt des Akkumulators positiv, negativ oder null ist. Solche Befehle dienen z.B. zum Vergleich zweier Zahlen, um festzustellen, ob die eine größer als die andere ist. Ein Beispiel einer Verzweigung ist in Bild 5.13 dargestellt.

Unterprogrammsprünge
(Bild 5.14)

Mit einem bedingten Unterprogrammsprung werden – wie Bild 5.14 zeigt – Sprünge zu verschiedenen Unterprogrammen durchgeführt. Nachdem das angesprochene Unterprogramm vollständig abgearbeitet worden ist, folgt ein Sprung zurück zum Hauptprogramm. Die Steuerung des Prozessors wird dann wieder von diesem Programm übernommen.

Interrupts
(Bild 5.15)

Interrupts (Unterbrechungen) sind ein Hilfsmittel, um gewisse externe Ereignisse und interne Prozessor-Ereignisse bzw. Zustände zu erkennen (Bild 5.15).

Bearbeitung von Interrupts
(Bild 5.16)

Wenn ein Eingabe/Ausgabegerät bereit ist, Daten an den Prozessor zu übertragen, kann diese Bereitschaft durch ein Interruptsignal, entsprechend Bild 5.16, angezeigt werden. Dieses Unterbrechungssignal veranlaßt den Prozessor mit seiner laufenden Ausführungssequenz zu unterbrechen, wenn der gerade laufende Maschinenbefehl vollständig ausgeführt ist. Danach erfolgt ein unbedingter Sprung zu einem vorher definierten Ort, wo der Interrupt abgearbeitet wird. Zur Ausführung dieser Prozedur verwendet man ein Register-Stack.

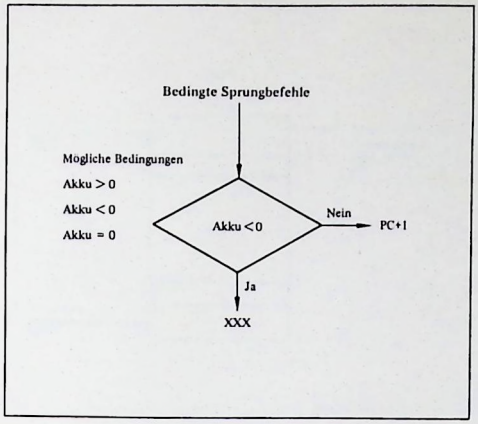


Bild 5.13
Bedingte Sprungbefehle

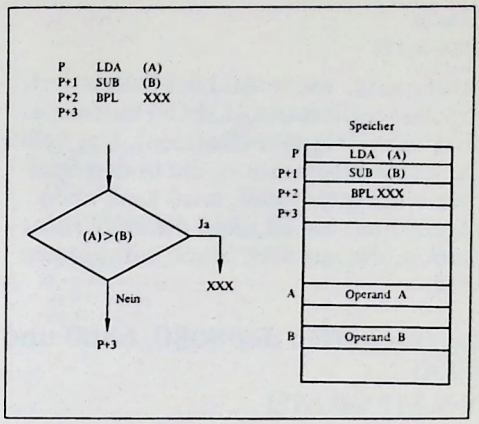


Bild 5.14
Bedingter Unterprogrammsprung

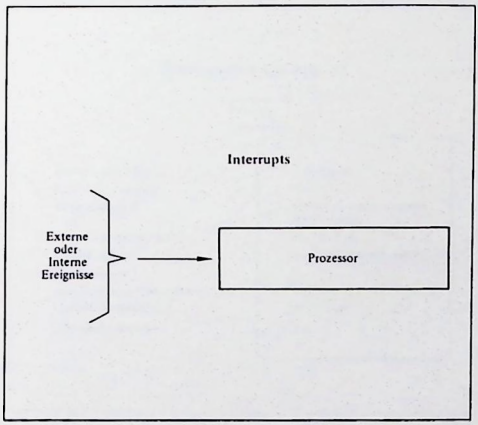


Bild 5.15
Interrupts

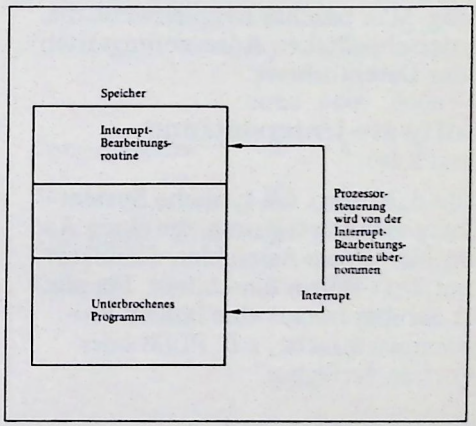


Bild 5.16
Interrupt-Bearbeitung

Stack
(Bild 5.17)

Stacks sind, wie in Bild 5.17 illustriert, sog. last-in/first-out (LIFO-) Pufferspeidher oder -Register (Dateien). Das heißt, die letzte Information, die in den Speicher eingespeist wird, wird ganz oben angeordnet. Sie ist dann die erste Information, die aus dem Stack entnommen wird.

Eigenschaften der 8080, 6800 und 9900
(Bild 5.18 und 5.19)

In den Bildern 5.18 und 5.19 sind einige Eigenschaften der 8080-, 6800- und 9900-Mikroprozessoren aufgeführt. Obwohl alle Modelle sehr ähnlich erscheinen, gibt es doch einige wesentliche Unterschiede hinsichtlich der Programmierung. Man beachte beispielsweise die unterschiedlichen Adressierungsarten oder Datenformate.

Software-Unterstützung
(Bild 5.20)

Bild 5.20 zeigt die typische System-Softwareunterstützung, die einen Assembler, Cross-Assembler, Simulator und Text-Editor einschließt. Für alle ist darüber hinaus eine höhere Programmiersprache, z.B. PL/M oder Fortran verfügbar.

Zusammenfassung

In dieser Lektion wurden allgemeine Softwareeigenschaften von Mikroprozessoren besprochen.

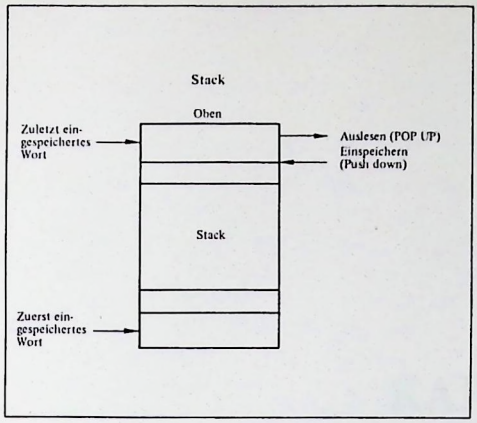
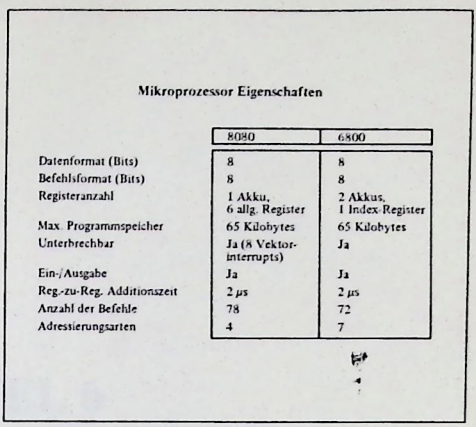
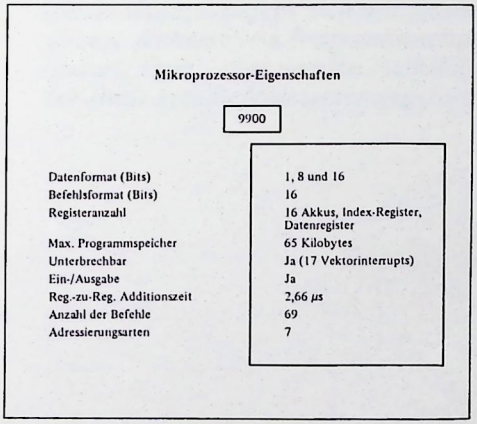


Bild 5.17
Stack



	8080	6800
Datenformat (Bits)	8	8
Befehlsformat (Bits)	8	8
Registeranzahl	1 Akku, 6 allg. Register	2 Akkus, 1 Index-Register
Max. Programmspeicher	65 Kilobytes	65 Kilobytes
Unterbrechbar	Ja (8 Vektorinterrupts)	Ja
Ein-/Ausgabe	Ja	Ja
Reg.-zu-Reg. Additionszeit	2 µs	2 µs
Anzahl der Befehle	78	72
Adressierungsarten	4	7

Bild 5.18
Eigenschaften des 8080 und 6800 Mikroprozessors



Datenformat (Bits)	1, 8 und 16
Befehlsformat (Bits)	16
Registeranzahl	16 Akkus, Index-Register, Datenregister
Max. Programmspeicher	65 Kilobytes
Unterbrechbar	Ja (17 Vektorinterrupts)
Ein-/Ausgabe	Ja
Reg.-zu-Reg. Additionszeit	2,66 µs
Anzahl der Befehle	69
Adressierungsarten	7

Bild 5.19
Eigenschaften des 9900 Mikroprozessors

	8080	6800	9900
Cross-Assembler	X	X	X
Simulator	X	X	X
Text Editor	X	X	X
Compiler	X	X	X

Bild 5.20
Software-Unterstützung

KAPITEL 6

BEFEHLSVORRAT

Dieses Kapitel befaßt sich mit festen und vom Anwender definierten Befehlssätzen. Anhand von Programmbeispielen werden diese zwei Befehlssätze erläutert. Dem Leser werden Richtlinien zur Auswahl von Mikroprozessoren auf der Basis von Befehlssätzen gegeben.

Befehlsvorrat

In diesem Kapitel wird auf feste und vom Anwender definierte Befehlssätze eingegangen. Es gibt einen Überblick über die Befehlssätze einiger gängiger Mikroprozessoren. Zum Schluß wird ein für den 8080 geschriebenes Programm untersucht und im Detail besprochen.

Mikroprozessor-Blockschaltung (Bild 6.1)

Um zu verstehen, wie ein Befehlssatz spezifiziert wird, muß man sich daran erinnern, daß sich ein Mikroprozessor aus bestimmten Blöcken bzw. Baugruppen zusammensetzt. Aus der Blockschaltung in Bild 6.1 geht hervor, wie die Funktionselemente auf einem Chip organisiert sein können. Als erstes entscheidet der Hersteller, welche Basis-Baugruppen er verwenden will und wie er diese miteinander verbindet. Danach definiert er eine Art Befehlssatz. Ein solcher Befehlssatz besteht aus einer Liste von Steuerwörtern, die die Art und Weise bestimmen, in welcher die Architektur-Blöcke für ein gegebenes Verbindungsmuster zusammenarbeiten.

Man unterscheidet zwei Arten von Befehlssätzen: feste und vom Anwender definierte. Unter einem festen Befehlssatz versteht man eine Liste, die der Hersteller dem Anwender übergibt. Feste Sätze ermöglichen gewöhnlich eine sehr effektive Anwendung eines gegebenen Mikroprozessors, dessen Baugruppen auf eine bestimmte Art zusammenwirken.

Kann der Anwender frei über den Befehlssatz bestimmen oder diesen selbst definieren, spricht man von einem vom Anwender definierten Befehlssatz. Um über Anwender-definierte Sätze verfügen zu können, muß der Mikroprozessor mikroprogrammierbar sein. Das hat allerdings nichts zu tun mit der Tatsache, daß einige Mikroprozessoren mikroprogrammiert sind. Denn ein mikroprogrammierter Mikroprozessor bedeutet nicht zugleich, daß er auch mikroprogrammierbar ist.

Gegenwärtig gibt es zwei wichtige Ebenen der Prozessor-Programmiertechniken, nämlich die Mikroprogrammierung und die Makroprogrammierung. Unglücklicherweise entstand das Wort Mikroprogrammierung vor der Entwicklung der Mikroprozessoren. Deshalb verstehen viele Entwickler unter Mikroprogrammierung die Programmierung eines Mikroprozessors. Das ist allerdings nicht richtig. Denn ein Mikroprogramm ist eine Art Programm, das direkt die Arbeitsweise eines jeden Funktionselementes in einem Mikroprozessor steuert, Aus diesem Grund ist nicht jedes Programm für einen Mikroprozessor ein Mikroprogramm.

Grundsätzlich wird in einer mikroprogrammierbaren Maschine die Logik-Konfiguration in der arithmetisch/logischen Einheit (ALU) und in den damit verwandten Untergruppen durch eine Reihe von Befehlen gesteuert. Diese Befehle werden Mikrobefehle (Mikroinstruktionen) genannt. Ein Mikrobefehl wird durch ein Bitmuster gebildet, das im Mikroprogramm-Speicher (entweder ROM oder PROM) gespeichert ist.

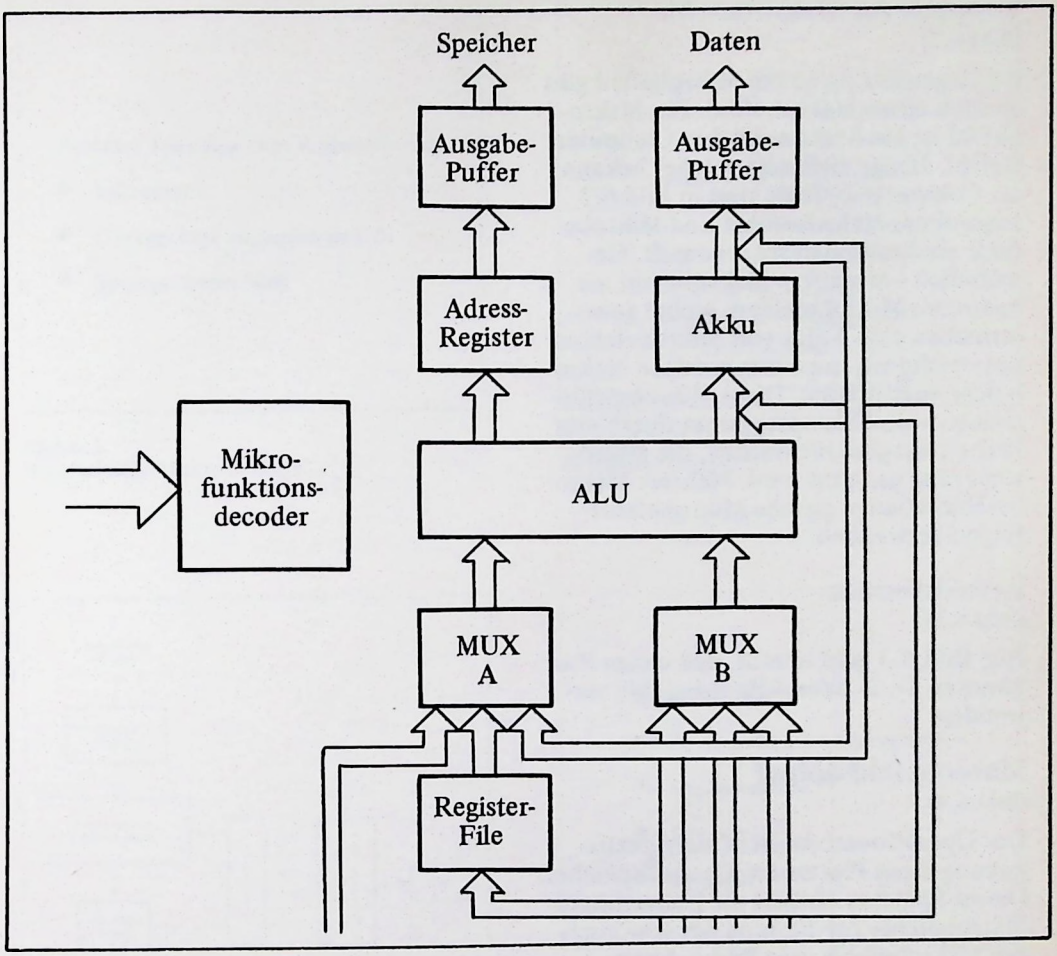


Bild 6.1
Mikroprozessor-Blockdiagramm

Beispiele für Makrobefehle
(Bild 6.2)

Im Gegensatz zu einem Mikrobefehl gibt es auch einen Makrobefehl. Ein Makrobefehl ist ein herkömmlicher Computer-Befehl. Einige Beispiele solcher bekannter Computer-Befehle sind in Bild 6.2 angegeben. Makrobefehle und Mikrobefehle sind miteinander verwandt. Gewöhnlich besteht ein Makrobefehl aus mehreren Mikrobefehlen, wobei gewissermaßen eine Folge von Mikrobefehlen notwendig ist, um einen einzigen Makrobefehl auszuführen. Diese Mikrobefehle müssen vom Mikroprogramm durch eine Einheit ausgewählt werden, die Mikrosteuerung genannt wird. Mehrere Makrobefehle können gleiche Mikrobefehlsfolgen verwenden.

Befehlsformat
(Bild 6.3)

Aus Bild 6.3 geht hervor, daß einige Prozessoren 1-, 2- oder 3-Byte-Befehle verwenden.

Makrobefehlsablauf
(Bild 6.4)

Der Operationscode dient zur Spezifizierung eines Platzes im Kontrollspeicher. Dieser Speicher enthält die notwendigen Mikrobefehle zur Bildung des gewünschten Makrobefehls, wie Bild 6.4 zeigt. Da einige Mikroprozessoren mikroprogrammierbar sind, verfügen diese über eine beträchtliche Flexibilität. Insbesondere können solche Mikroprozessoren den Befehlssatz von anderen Mikroprozessoren emulieren.

Befehlsklassen
(Bild 6.5)

Die Anweisungen (statements) der meisten Befehlssätze können in vier Klassen unterteilt werden. Sie sind in Bild 6.5 angegeben.

Addiere Speicher zum Registerinhalt

- Inkrement
- Überspringe nächsten Befehl
- Springe wenn Null

Bild 6.2
Beispiele für Makrobefehle

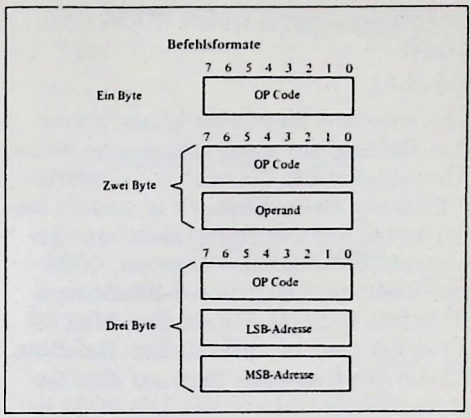


Bild 6.3
Befehlsformate

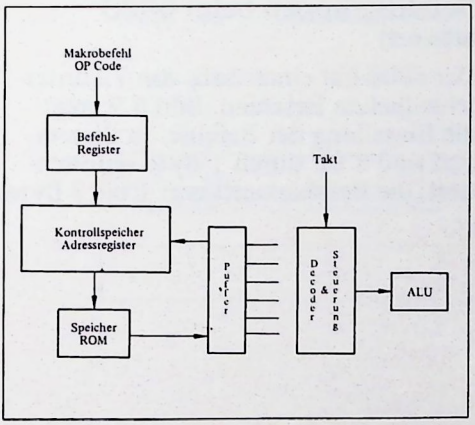


Bild 6.4
Makrobefehlsablauf

Befehlsklassen

- Arithmetische und logische
- Datentransfer
- Springe
- Spezielle

Bild 6.5
Befehlsklassen

Befehlsgruppen beim 4004 und 4040
(Bild 6.6)

Verschiedene Hersteller klassifizieren ihre Befehle auf unterschiedliche Weise. Allerdings sollte die vorher definierte Einteilung einen Einblick in andere Befehlssätze auf der Funktionsebene gewähren. Die Mikroprozessoren 4004 und 4040 sind parallele 4-Bit-Zentraleinheiten (CPUs). Mit 46 Befehlen für die 4004 und 14 zusätzlichen Befehlen für die 4040 kommt man auf eine Gesamt-Befehlszahl von 60. Die 4004 ist mit der 4040 in der Weise aufwärtskompatibel, daß ein für die 4004 geschriebenes Programm auch auf der 4040 läuft. Zusätzlich zu den 16 Registern der 4004 kommen bei der 4040 noch 8 Indexregister hinzu. In Bild 6.6 sind die wichtigsten Befehlsgruppen dargestellt.

Die ersten drei Gruppen sind beiden Prozessoren gemeinsam, während die vierte ausschließlich für die 4040 vorgesehen ist. Ein Blick auf die vollständige Befehlsliste zeigt, daß die Gruppe der Maschinenbefehle sowohl viele arithmetische und logische Anweisungen als auch Sprungbefehle umfaßt. Die Eingabe/Ausgabebefehle beziehen sich hauptsächlich auf den Datenaustausch zwischen dem Akkumulator und dem RAM's oder ROM's. Die Akkumulator-Befehlsgruppe umfaßt Funktionen, die allein mit dem Akkumulator ausgeführt werden.
Es sind dies z.B. Funktionen wie lösche Akkumulator, oder Übertrags- und Inkrementbildung usw. Die 4040-Gruppe steuert die zusätzlichen Indexregister und die Interruptmöglichkeiten des 4040.

Beispiel eines 4004/4040-Befehls
(Bild 6.7)

Bild 6.7 zeigt ein Beispiel eines Maschinenbefehls für die 4004 und 4040. Beim Beispiel handelt es sich um einen Ein-Wort-Befehl mit dem symbolischen Code INC. Das bedeutet: Inkrementiere den Inhalt eines Registers. Der Binärcode besteht aus zwei 4-Bit-Gruppen. Die erste Gruppe mit den vier Bits 0110 enthält den Operationscode. Die zweite Gruppe der vier Bits, RRRR, stellt die Adresse eines der 16 Register in der CPU dar.

JUN-Befehl
(Bild 6.8)

Manche Befehle der 4004/4040 erfordern zwei Speicherworte. Solche 2-Byte-Befehle sind in Bild 6.8 wiedergegeben. Diese Zwei-Wort-Befehle besetzen zwei aufeinanderfolgende Speicherplätze in einem ROM. Zum Beispiel: JUN bedeutet einen unbedingten Sprung zum ROM mit der Adresse A_3, A_2, A_1.

Befehlsgruppen beim 6800
(Bild 6.9)

Der 6800 hat einen Satz von 72 unterschiedlichen Befehlen. Bild 6.9 zeigt die Einteilung der Befehle. In diesem Fall sind 8 Bit durch 1 Byte repräsentiert; die Befehle umfassen 1 bis 3 Bytes.

- Maschine
- Ein-/Ausgabe-Gruppe
- Akkumulator-Gruppe
- 4040-Gruppe

Bild 6.6
4004 und 4040 Befehlsgruppen

Mnemonic Code: INC
Op Code: 0110
Adresse: RRRR

Bild 6.7
4004/4040 Befehlsbeispiel

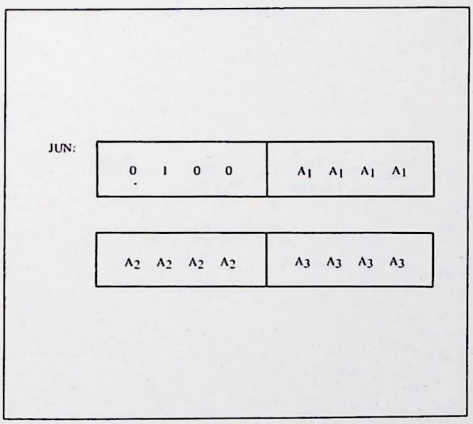


Bild 6.8
4004/4040 JUN-Befehl

Befehlsgruppen
- Akkumulator und Speicher
- Index Register und Stack
- Bedingte/unbedingte Sprünge
- Statusregister

Bild 6.9
6800 Befehlsgruppen

6800-Operationscode
(Bild 6.10)

Bei der Betrachtung der Codes für den 6800 stellt man fest, daß er sieben Adressierungsarten besitzt. Die verschiedenen Operationscodes hängen von der gewünschten Adressierungsart ab. Als Beispiel bilde man die UND-Funktion der Inhalte des Akkumulators A und des Speichers M und bringe das Ergebnis nach A. Bild 6.10 zeigt die Operationscodes für jede Adressierungsart. Die Codes liegen im hexadezimalen Format vor.

Befehlsgruppen beim PPS-4
(Bild 6.11)

In Bild 6.11 sind die Befehlsgruppen für die CPU PPS-4 dargestellt. Dieser Mikroprozessor benützt 4-Bit-Datenworte und 8-Bit-Befehlsworte. Der Befehlssatz umfaßt 50 feste Befehle.

Dezimal-Korrektur beim PPS-4
(Bild 6.12)

Der PPS-4-Befehl für die Dezimal-Korrektur wird aus einem Mnemonic-Code von DC und einem Hexadezimal-Code von 65 gebildet (Bild 6.12). Der dazugehörige Dualcode ist direkt unter den Hexadezimal-Code geschrieben. Dieser Befehl addiert die Dualzahl 1010 zum Inhalt des Akkumulators. Das Ergebnis wird im Akkumulator A gespeichert.

Hinweise zu Befehlssätzen
(Bild 6.13)

In Bild 6.13 sind einige Merkmale fester Befehlssätze zusammengefaßt. Die arithmetischen Befehle sind üblicherweise für folgende Aufgaben vorgesehen: Addition, Addition mit Übertrag (Carry), Subtraktion und Subtraktion mit Borgen (Borrow). Logische Anweisungen bestehen aus Booleschen Gleichungen (z.B. UND, ODER usw.). Datentransport-Anweisungen sind beispielsweise: Tausche Inhalte von Akkumulator und Speicher aus, lade Akkumulator usw. Als Verzweigungs-Anweisungen kommen Übersprünge bei Übertrag (Skip), ferner springe bei null, springe unbedingt usw. in Frage.

Befehl	Adressierungsart			
	IMMD	DIR	INDEX	EXTND
ANDA:	84	94	A4	B4

Bild 6.10
6800 Befehlscode

Befehlsgruppen

- Arithmetische und logische
- Transporte
- Unterprogrammsprünge
- RAM-Adressierung
- Ein-/Ausgabe

Bild 6.11
PPS-4 Befehlsgruppen

DC: 6 5
0110 0101

A + 1010 → A

Bild 6.12
PPS-4 Dezimalkorrektur

µ-Prozessor	Arith-Log.	Trans-fer	Sprünge	Spezial	Gesamt
4004	4	11	6	19	40
4040	4	11	6	39	60
6800	19	10	18	25	72
PPS-4	10	24	11	5	50

Bild 6.13
Befehlssätze

Blockschaltung des 3002
(Bild 6.14)

Die nächsten zwei Mikroprozessoren verfügen über Anwender-definierte Befehlssätze. Sie gehören zu der neueren Mikroprozessor-Gruppe mit der sog. "Bit slice"-Architektur. Der erste ist der 3002, der zweite der SBP0400. Die Blockschaltung des Modells 3002 ist in Bild 6.14 dargestellt. Für die Programmspeicherung benützt der 3002 bipolare Standard-Festwert- und Schreib-/Lesespeicher. Die programmierbaren (PROM's) und maskenprogrammierbaren Festwertspeicher (ROM's) werden zur festen Speicherung der Mikrobefehle verwendet. Zur Speicherung von Mikrobefehlen werden aber auch Schreib-/Lesespeicher (RAM's) benützt; allerdings nur in der Programm-Entwicklungsphase, um die Fehlersuche und -Korrektur zu vereinfachen.

Mikrofunktions-Decoder des 3002
(Bild 6.15)

Beim 3002 wurde eine andere Terminologie eingeführt als bei festen Befehlssätzen üblich – und zwar wird hier der Ausdruck "Mikrofunktion" verwendet. Einzelheiten des Mikrofunktions-Decoders sind in Bild 6.15 dargestellt. Eine zu erzeugende Mikrofunktion wird einerseits durch die sog. Funktions-Gruppe (F-Gruppe) und andererseits durch die Register-Gruppe (R-Gruppe) bestimmt, die beide durch den Decoder ausgewählt werden. Die F-Gruppe wird durch die oberen drei Bits der Daten F_4 - F_6 spezifiziert, während die R-Gruppe durch die unteren vier Bits definiert ist.

K-Bus und I-Bus des 3002
(Bild 6.16)

Mit den acht Bedingungen in der F-Gruppe und den drei Bedingungen in der R-Gruppe erhält man 24 Basis-Gruppen. Wie in Bild 6.16 verdeutlicht, können sich für jede dieser Gruppen andere Bedingungen ergeben, und zwar als Folge der Mikrofunktion mit dem Zustand (entweder 00 oder 11) der K-Bus-Eingänge.

3002-Additionsablauf
(Bild 6.17)

Bild 6.17 zeigt, wie ein Addierbefehl durch eine Mikrofunktion im 3002 erzeugt bzw. implementiert werden kann. Während des Addiervorgangs sind zwei wesentliche Mikrofunktionen erforderlich. Erstens, der zu addierende Operand muß gelesen werden; zweitens, die Addition muß ausgeführt werden. Der erste Teil erfolgt durch die Übertragung der Adresse des Operanden vom Mikrobefehls-Wort (die zwei niederwertigsten Bytes) zum Speicher-Adressenregister. Dies läßt sich mit Hilfe des Befehls LMI bewerkstelligen. Diesen Befehl findet man in der F-Gruppe 1, R-Gruppe 1, in Verbindung mit dem auf 00 gesetzten K-Bus. Als nächstes wird der Operand vom I-Bus gelesen und zum Inhalt des Akkumulators plus Übertrag addiert. Das Ergebnis wird in den Akkumulator A zurückgebracht.

Symbolisch läßt sich dies in der Form I+AT+CI übertragen nach AT schreiben. Hierbei bedeutet AT Akkumulator oder T-Register. Die Ausführung kann mit dem Befehl AIA erfolgen, der in der F-Gruppe 3, R-Gruppe 3 mit dem K-Bus auf 11 zu finden ist.

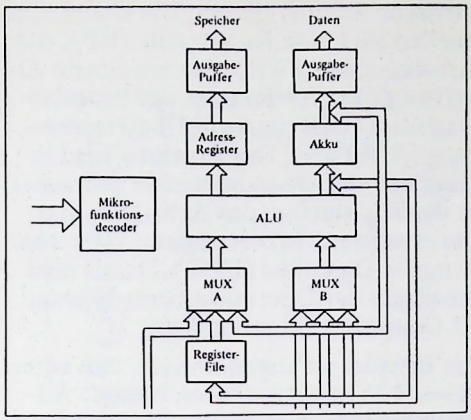


Bild 6.14
3002 Blockdiagramm

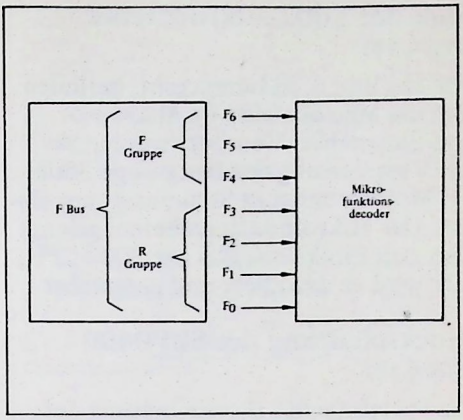


Bild 6.15
3002 Mikrofunktionsdecoder

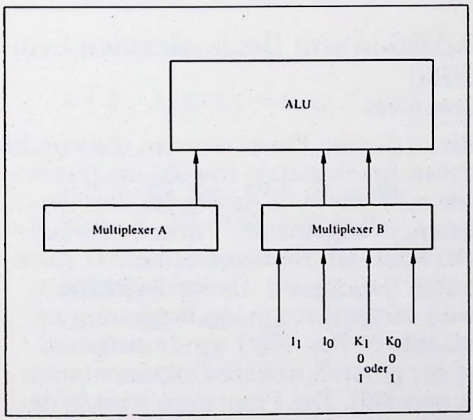


Bild 6.16
3002 K-Bus und I-Bus

LMI

F – Gruppe 1 R – Gruppe 1

K – Bus = 00

A1A

F – Gruppe 3 R – Gruppe 3

K – Bus = 11

Bild 6.17
3002 Additionsablauf

Fluß der 3002-Mikrobefehle
(Bild 6.18)

Wie aus Bild 6.18 hervorgeht, befinden sich die Mikrobefehle im Mikroprogrammspeicher. Von dort werden sie auf Veranlassung der Baugruppe 3001, der Mikroprogramm-Steuerung, ausgelesen. Der Mikrofunktions-Befehl gelangt über den Funktions-Bus zur 3002-CPU; hier wird er decodiert und ausgeführt.

Blockschaltung des SBP0400
(Bild 6.19)

Eine vereinfachte Blockschaltung des SBP0400 zeigt Bild 6.19. Ein externes 9-Bit-Steuerwort erzeugt 512 Ein-Takt-Operationen. Die Speicherung dieser Programmwörter kann ebenso wie beim 3002 in einem bipolaren Speicher erfolgen. Der 0400 besitzt einige zusätzliche Eigenschaften, die seine Leistungsfähigkeit erhöhen. So verfügt er über eine vorausschauende Befehlseinrichtung, wodurch sich Befehlsabruf und Ausführungszyklen überlappen können. Ferner sind eigene Anschlüsse für die Stellenkennzeichnung vorgesehen, um die Verarbeitung von zeichenbehafteten Daten bei Multichip-Aufbauten zu ermöglichen. Und andere Anschlüsse erlauben eine kontinuierliche Operanden-Überwachung bei datenabhängigen Entscheidungen außerhalb des Chips.

Operationsform II und Mikrobefehle des SBP0400
(Bild 6.20)

Der 0400 verfügt über Operationsformen und Mikrobefehle. Es gibt sechs Operationsformen. Insgesamt sind in der PLA (programmierbares logisches Array) 512 Ein-Takt-Operationen programmiert. Wie beim 3002 betrachten wir die Programmierung der Addition.

Die Operationsform II ist im oberen Teil von Bild 6.20 erläutert. Normalerweise kann man mit der Operationsform II von 2 der 4 Operandenquellen die arithmetische Summe bilden. Die Operandenquellen sind eine Registerfile (RF), das Arbeitsregister (WR), das erweiterte Arbeitsregister (XWR) oder der Eingabe-Datenbus (DIB) und ein Übertragseingang (ALUCIN). Das Ergebnis wird in eines der vier Operandenziele gebracht: in die Registerfile, das Arbeitsregister, das erweiterte Arbeitsregister oder zum Ausgabe-Datenbus (DOB). Damit sind innerhalb der Operationsform II bis zu 12 Operationsarten möglich.

Als Beispiel sei angenommen, daß einer dieser 12 Operationsarten besagt: Addiere die Information des Eingabe-Datenbus (DIB) zum Inhalt des Arbeitsregisters (WR) plus dem Eingangs-Übertrag (ALUCIN) und bringe das Ergebnis auf den Ausgabe-Datenbus (DOB). Diese Mikrooperation-Addition ist im unteren Teil von Bild 6.20 dargestellt.

Addition von Dezimalzahlen beim 8080
(Bild 6.21)

Bis zu diesem Punkt wurden die verschiedenen Befehlssätze, sowohl die festen wie auch die vom Anwender definierbaren, in allgemeiner Form behandelt. Der 8080-Mikroprozessor besitzt einen festen Befehlssatz. Dieser Befehlssatz wird verwendet um ein Programm zu schreiben. Der 8080 wurde aufgrund seiner guten Software-Dokumentation ausgewählt. Das Programm wird in der 8080-Assemblersprache geschrieben, da sie auf dem 8080-Befehlssatz aufgebaut ist. Nachfolgend wird gezeigt, wie man mit einem Befehlssatz arbeitet, was man beim Schreiben von Mikroprozessor-Programmen beachten muß und wie die Eigenschaften eines Mikroprozessors den Wirkungsgrad des Programmes bei einigen Anwendungen beeinflußen. Damit erhält man zugleich einen besseren Einblick, wenn es darum geht, den richtigen Chip-Satz für eine gegebene Anwendung auszusuchen.

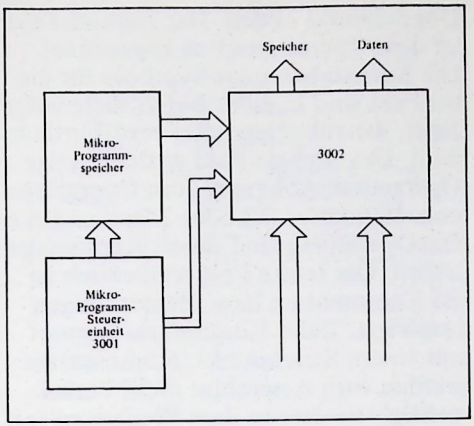


Bild 6.18
3002 Mikrobefehlsablauf

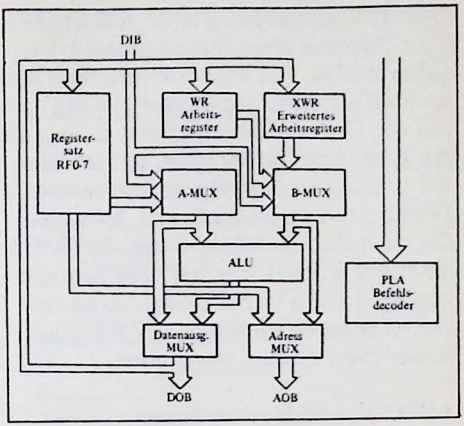


Bild 6.19
SBP0400 Blockdiagramm

A + B + ALUCIN → A od. B od. C

wobei:

A.B ist RF, WR, XWR oder DIB
C ist RF, WR, XWR oder DOB

Mikrobefehl:

DIB + WR + ALUCIN → DOB

Bild 6.20
SBP0400 Befehlsform II

```
  521493
+617380
-------
 1138873

BEGIN:  MVI   C,3       ; C = NO OF WORDS
        LXI   D,VAR 1   ; D AND E ADDR OF VAR 1
        LXI   H,VAR 2   ; H AND L ADDR OF VAR 2
        XRA   A         ; CLEARS CARRY BIT
LOOP:   LDAX  D         ; LOAD BYTE OF VAR 1
        ADC   M         ; ADD BYTE OF VAR 2
                        ; WITH CARRY
        DAA             ; DECIMAL ADJUST
        STAX  D         ; STORE RESULTS IN VAR 1
        DCR   C         ; C = C - 1
        JZ    DONE
        INX   D         ; POINT TO NEXT BYTE OF VAR 1
        INX   H         ; POINT TO NEXT BYTE OF VAR 2
        JMP   LOOP      ; ADD NEXT TWO BYTES
VAR 1:  DB    93H
        DB    14H
        DB    52H
VAR 2:  DB    80H
        DB    73H
        DB    61H
DONE:   HLT
        END
```

Bild 6.21
Addition dezimaler Zahlen mit dem 8080

Das vorliegende Programm soll die dezimale Summe von zwei dezimalen Variablen bilden. Bei Prozessoren ohne diese Vielfach-Arithmetikeinrichtung müssen die Dezimalzahlen zuerst in Dualzahlen umgewandelt werden, um sie nach der arithmetischen Operation wieder in Dezimalzahlen umzusetzen. Es sollen die zwei Zahlen 521493 und 617380 zueinander addiert werden; ihre Summe beträgt 1138873. In Bild 6.21 ist ein vollständiges Programm zur Addition dieser zwei Zahlen angegeben.

8080-RAM-Darstellung
(Bild 6.22)

Die zwei Zahlen 521493 und 617380 bestehen jeweils aus sechs Dezimalziffern. Da zur Darstellung jeder Dezimalziffer vier Bits erforderlich sind, die 8080-Struktur jedoch 8-Bit-Worte erlaubt, können pro Wort – wie in Bild 6.22 gezeigt – jeweils zwei Ziffern zusammengefaßt werden.

8080-Registerstruktur
(Bild 6.23)

Um die Werte der Variablen in Bild 6.23 zu speichern, werden sechs Worte oder Speicherplätze in einem externen 8-Bit-organisierten Datenspeicher (RAM) benötigt. Zusätzlich zum externen Datenspeicher besitzt der 8080 einen 8-Bit-Akkumulator und sechs allgemeine 8-Bit-Datenregister.

8080-Befehlsfelder
(Bild 6.24)

Wie aus Bild 6.24 ersichtlich ist, besteht in der 8080-Assemblersprache ein Befehl aus vier Feldern oder Teilen. Zwischen jedes Feld eines Befehls ist ein Zwischenraum bzw. Leerzeichen eingefügt. Das erste Feld ist frei wählbar und wird Kennzeichnungsfeld (Label-Feld) genannt. Es besteht der Länge nach aus eins bis fünf Zeichen und muß mit einem Buchstaben beginnen und mit einem Doppelpunkt enden. Das nächste Feld ist dem Operationscode zugeordnet. Die Buchstaben oder Symbole für dieses Feld sind in einer Befehlsliste aufgeführt, die vom Hersteller veröffentlicht wird. Das nächste Feld enthält einige Operanden. Abhängig vom Operationscode können es 0,1 oder 2 Operanden sein. Die Operanden sind durch Kommas getrennt. Das letzte Feld schließlich ist für Kommentare bzw. Bemerkungen reserviert. Jeder Kommentar beginnt mit einem Strichpunkt. Kommentare werden vom Assembler nicht berücksichtigt; sie dienen dem Programmierer zur Erklärung bzw. Erinnerung.

8080-Befehlsdetails
(Bild 6.25)

Um zu beschreiben, was während der Ausführung des Programms stattfindet, erinnere man sich an das dezimale Additionsprogramm für eine Maschine, die nur duale Arithmetik bewältigt. Allerdings kann jede 4-Bit-Zahl so lange als Dezimalzahl behandelt werden, so lange sie die Ziffern 0 bis 9 repräsentiert. Es gibt jedoch eine Einschränkung: Das Bit-Muster kann nicht auch die Darstellung der hexadezimalen Ziffern A bis F umfassen. Damit bei der Ausführung einer Addition die dezimale Deutung erhalten bleibt, muß zu einer 4-Bit-Zahl der Wert sechs addiert werden, wenn die Addition ein Ergebnis zwischen 10 und 15 liefert. Der Grund liegt darin, daß eine 4-Bit-Zahl sechs mehr Bit-Kombinationen darstellen kann als es Dezimalziffern gibt.

Dadurch entsteht jedoch ein Zehnerübertrag der 4-Bit-Zahl. Deshalb enthält das Programm einen Befehl für die Dezimalkorrektur (DAA). Die Addition erfolgt nun in der Weise, daß zuerst die Summe der zwei niedrigwertigen Bytes plus Übertrag gebildet und dann die Dezimalkorrektur durchgeführt wird. Der gesamte Vorgang wiederholt sich so lange, bis alle drei Gruppen addiert sind.

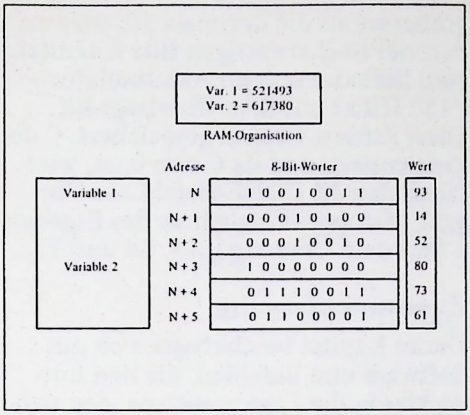


Bild 6.22
8080 RAM-Organisation

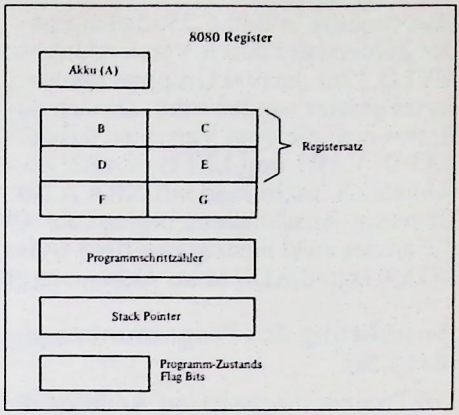


Bild 6.23
8080 Registerstruktur

Label-feld	OP Code	Operanden	Bemerkungen
Beginn:	MVI	C,3	; C = No. of words

Bild 6.24
8080 Befehlsfelder

```
Beginn:   MVI   C,3
          LXI   D, VAR1
          LXI   H, VAR2
          XRA   A
```

Bild 6.25
8080 Befehlsdetails

Man beachte in Bild 6.25, daß zuerst der Zählerstand durch Verwendung von MVI C,3 für die drei Gruppen der zwei Bytes gesetzt werden muß. Danach definiert man die zwei Variablen durch LXI D, VAR1 und LXI H, VAR2; schließlich löscht man mit XRA A den Übertrag. Anschließend beginnt die Addition der zwei niedrigstwertigen Bytes LDAX D und ADC M im Akkumulator.

Ausführung des Programmbeispiels (Bild 6.26)

Das Programmbeispiel zur Addition der Zahlen 521493 und 617380 ist in Bild 6.26 dargestellt.

1. Additionsschritt: Der Akkumulator enthält nach der ersten Addition eine 13 (hexadezimal); der Übertrag ist 1 und der Zwischenübertrag am Additionsende der ersten vier Bits ist 0.

Dezimalkorrektur: Führt den Dezimalkorrektur-Befehl DAA aus. Da das Übertrags-Bit gesetzt ist, wird zu den vier höchstwertigen Bits die Zahl 6 addiert. Dies ergibt im Akkumulator "73" in hexadezimaler Form. Nun speichert man diese Ziffern (mit STAX D, Bild 6.21) und dekrementiert C (Schleifenzähler) sowie die Adressregister DE und HL.

2. Additionsschritt: Die nächsten zwei Gruppen, zusammen mit dem bei der letzten Dezimalkorrektur erhaltenen Übertrag, werden addiert. Das Ergebnis im Akkumulator beträgt jetzt 88 in Hex-Form.

3. Additionsschritt: Wiederum wird der DAA-Befehl durchgeführt. Es erfolgt jedoch keine Addition, da jede Ziffer kleiner ist als die dezimale 10. Nun speichert man die zwei Ziffern und dekrementiert C, DE und HL um die zwei letzten Bytes der Variablen 1 und 2 zu adressieren. Im Anschluß daran werden die letzten zwei Gruppen addiert.

Dezimalkorrektur: Abermals wird der DAA-Befehl durchgeführt. Da "B" (Hex) größer ist als die dezimale 10, wird zu den vier höchstwertigen Bits 6 addiert. Nun befindet sich im Akkumulator "13" (Hex) mit dem Übertrags-Bit. Diese Ziffern werden gespeichert, C dekrementiert, und da C nun 0 ist, wird durch den JZ DONE-Befehl der Vorgang gestoppt. Folglich ist das Ergebnis 1 (aus dem Übertrag), 13, 88 und 73.

Zusammenfassung

Dieses Kapitel beschäftigte sich mit Software und Befehlen, die den Entwickler in die Lage versetzen, den richtigen Befehlssatz auszuwählen und für seine Aufgabe den Programmieraufwand abzuschätzen. Software ist das Werkzeug zur Programmierung eines Rechners. Andererseits sind Befehle die Bausteine, aus denen sich die Software zusammensetzt.

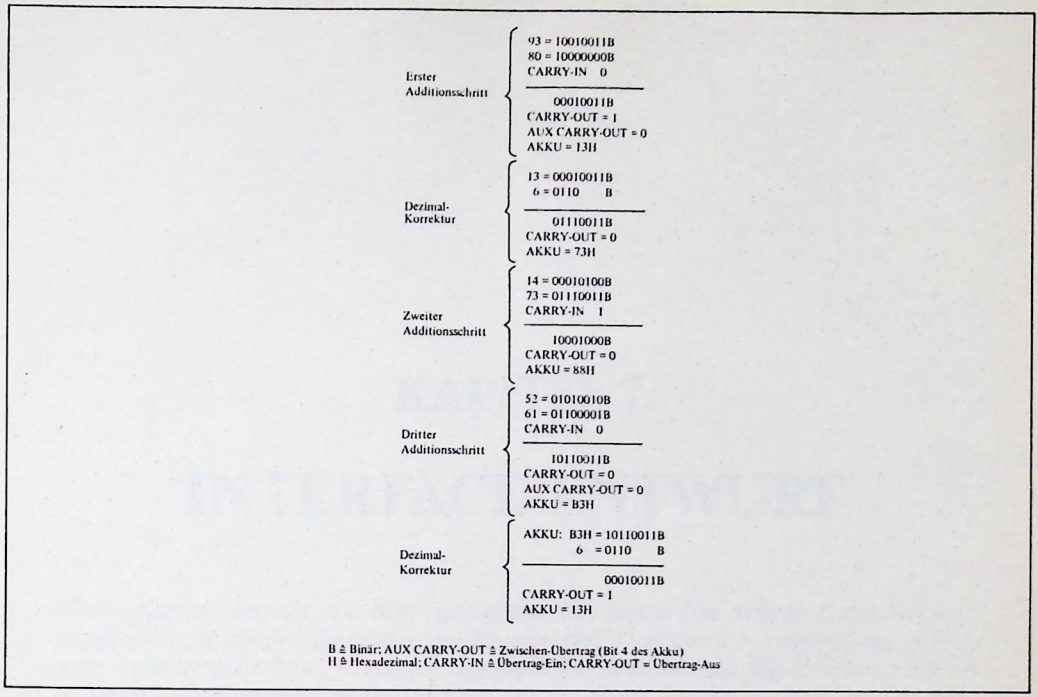


Bild 6.26
Ausführung des Programmbeispiels

KAPITEL 7

INTERFACE-ENTWURF

Dieses Kapitel befaßt sich mit Problemen, die durch den Anschluß des Mikroprozessors an andere Bausteine zu Tage treten. Den Ort der Verbindung nennt man Schnittstelle bzw. Interface. Deshalb bezeichnet man Bauelemente, die an solchen Schnittstellen zu finden sind, als Interface-Bausteine. Hier werden verschiedene Kategorien von Problemen besprochen und anhand spezifischer Beispiele nach Lösungen gesucht. Bei diesen Lösungen wird besonderer Wert auf die Verwendung von Standard-Bausteinen gelegt.

Interface-Entwurf

Der Inhalt dieses Kapitels sind der Interface-Entwurf und die damit verbundenen Problemlösungen. Da es sehr zahlreiche Interface-Situationen gibt, werden nachfolgend zuerst die Mikroprozessor-Interfaceprobleme aus der Sicht der Signalaufbereitung angegangen. Dann werden Lösungen erarbeitet, denen erhältliche Standard-Bausteine zugrunde liegen.

Interface-Betrachtungen
(Bild 7.1)

Interface-Probleme ergeben sich vor allem durch die unterschiedlichen Signalarten. Mit den in Bild 7.1 angegebenen Kategorien wird gewöhnlich jedes Interface erfaßt. Darunter befinden sich der Datenbus, der Adressbus, die Interrupt-Struktur, die Taktung und schließlich die verschiedenartigen Steuersignale.

Allgemeines Mikroprozessor-Interface
(Bild 7.2)

Um zu verstehen, wie dieses in einem System arbeitet, muß man sich daran erinnern, daß ein typischer Mikroprozessor aus den in Bild 7.2 gezeigten Grundelementen besteht. Dieses Bild gibt einen allgemeinen Fall wieder, der tatsächlich verwendete Mikroprozessor kann sich durch andere Eingangs- und Ausgangsleitungen unterscheiden. Gewöhnlich verfügt er über einen Adressbus, einen Datenbus, einen Kontrollbus und andere verschiedenartige Leitungen. Manche Mikroprozessoren verwenden einen einzigen bidirektionalen Bus für Daten sowie einige getrennte Eingangs- und Ausgangsbusse für die Geschwindigkeitsanpassung und zur Vereinfachung der Schnittstellen.

Interface-Probleme
(Bild 7.3)

Unabhängig von den verwendeten Bus-Techniken rühren die Probleme vom Umsetzen der Pegel, Treiben der Leitungen (Datenübertragung), der Pufferung und Datenumsetzung her. Diese in Bild 7.3 angegebenen Probleme sind wichtig, da sie uns während der gesamten Interface-Diskussion begegnen. Wie man diese Probleme erkennt und löst, ist der Zweck dieses Kapitels.

Port-Adressierung
(Bild 7.4)

Bild 7.4 zeigt, was mit einem Adressbus oder mit dem dafür vor kurzem aufgekommenen Ausdruck "Port-Adressierung" gemeint ist. Es sei angenommen, daß die CPU ein Datenwort zum Speicher oder zu irgendeinem E/A-Baustein sendet. Aber welcher ist der richtige Baustein? Zu dessen Auswahl muß es einen Weg geben. Wenn man genau wie in Speichern, Adressleitungen verwendet, kann ein E/A-Port (oder Baustein) von einer Gruppe so ausgewählt werden, daß nur er während eines bestimmten E/A-Zyklus angesprochen wird. Die ausgesuchte Mikroprozessor-Familie kann diese Funktion auf einem Speicher-Interfacechip oder einem E/A-Chip enthalten. Manche Familien verfügen über zwei Adressbusse: den einen für den Speicher und den anderen für die E/A-Bausteine.

- Datenbus
- Adreßbus
- Interrupts
- Taktung
- Steuersignale

Bild 7.1
Interface-Betrachtungen

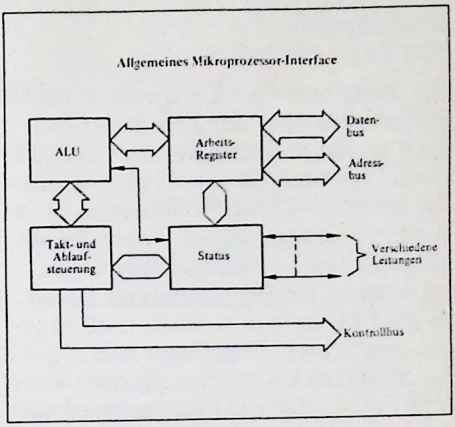


Bild 7.2
Allgemeines Mikroprozessor-Interface

- Pegel-Verschiebung
- Datenübertragung
- Pufferung
- Datenumsetzung

Bild 7.3
Interface-Probleme

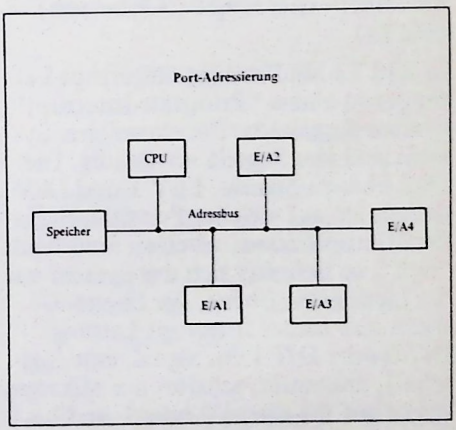


Bild 7.4
Port-Adressierung

Interrupts
(Bild 7.5)

Die dritte Interface-Kategorie ist die Unterbrechung bzw. der Interrupt (Bild 7.5). Interrupt-Techniken werden sehr häufig in Daten-Eingabezyklen benützt. Die gebräuchlichste Verwendung ist in Echtzeit-Anwendungen, wobei der Mikroprozessor weitgehend ausgelastet wird. Wenn ein Peripheriegerät Daten an die CPU senden möchte, muß es sozusagen "durch Zeigefinger heben", d.h. durch Setzen eines Zeichens oder einer Interrupt-Leitung auf sich aufmerksam machen. Solange die CPU durch bestimmte Vorsichtsmaßnahmen keine Schwierigkeiten hat, die jeweiligen Geräte zu identifizieren, können an eine Leitung mehr als ein Gerät angeschlossen werden.

Prioritäts-Interrupt (3 Ebenen)
(Bild 7.6)

In Bild 7.6 sind mehrere Interrupt-Leitungen in einem "Prioritäts-Interrupt"-System dargestellt. Für ein solches System sind vier Signale notwendig. Die zwei Ebenen-Signale, LEV 1 und LEV 2, deuten an, auf welcher Prioritätsebene der Mikroprozessor arbeitet. Sind beide "null", so befindet sich das System auf der niedrigsten Ebene, der Ebene 3. Wenn nun an der Interrupt-Leitung INT 2 oder INT 1 ein Signal, eine logische 1, ankommt, schaltet der Mikroprozessor auf die Ebene 2 oder 1 und beginnt dort zu arbeiten. Wenn er seine Aufgabe beendet hat, kehrt er auf die Ebene 3 zurück.

Typische Taktschaltung
(Bild 7.7)

Die nächste Interface-Kategorie sind Taktsignale (Bild 7.7). Die Mikroprozessor-Spezifikationen des Herstellers lassen auf diesem Gebiet nur wenig Entwurfsspielraum zu. Als Minimum werden üblicherweise ein Taktgenerator und dazugehörige Treiberschaltungen benötigt. In manchen Fällen ist ein Mehrphasentakt notwendig und es sind unter anderem Logikschaltungen erforderlich, um Totzeiten zwischen Taktphasen, Sperrsignale oder Austastsignale zu erzeugen. Deshalb gebrauchen Takt-Interfaceeinheiten Untergruppen von folgenden Interface-Gebieten: Pegelumsetzung, Leitungstreiber, Pufferung und Signalumsetzung.

Typisches Steuersignal-Interface
(Bild 7.8)

Die letzte Interface-Kategorie für Signale ist gewissermaßen ein Auffangbecken mit der Bezeichnung Steuersignale. Hierunter fallen manchmal wenig, manchmal viele verschiedene Signale, die für den richtigen Ablauf der Befehle notwendig sind. Um dies zu demonstrieren, zeigt Bild 7.8 den 8080-Mikroprozessor als vereinfachtes Blockdiagramm. Die eingeklammerten Anschlüsse und die gekennzeichneten Steuersignale sind für Status und Befehl notwendig, um Aufgaben außerhalb des Chips erledigen zu können. Und je nach Mikroprozessor werden diese Signale wiederum in ihrem Pegel verschoben, gepuffert usw.

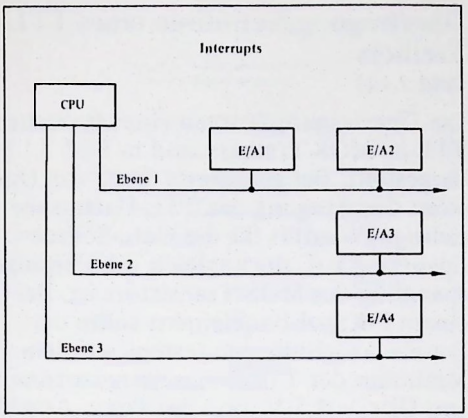


Bild 7.5
Interrupts

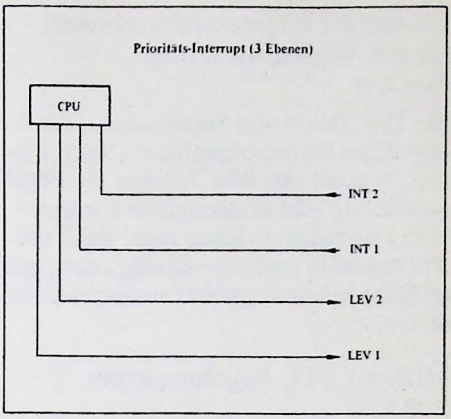


Bild 7.6
Prioritäts-Interrupt (3 Ebenen)

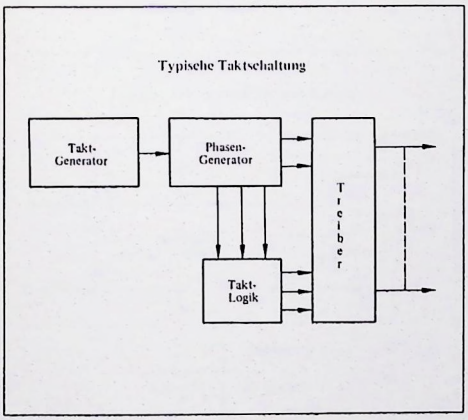


Bild 7.7
Typische Taktschaltung

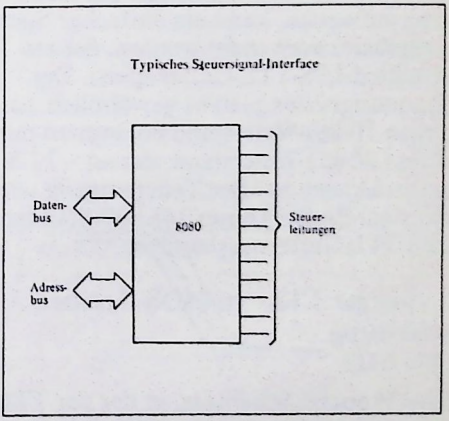


Bild 7.8
Typisches Steuersignal-Interface

Einfaches Widerstandsnetzwerk für die Pegelumsetzung
(Bild 7.9)

Die Diskussion von Schaltungen, die die jeweiligen Interfaceprobleme lösen können, beginnt mit Bild 7.9. Für die Pegelumsetzung gibt es zahlreiche annehmbare Lösungen; es kann dazu sogar der alte erprobte und zuverlässige, aber mangelhafte Spannungsteiler verwendet werden.

MOS zu TTL-Pegelumsetzer
(Bild 7.10)

Gegenüber der vorherigen Widerstandskopplung zeigt Bild 7.10 eine wesentlich wirkungsvollere Methode für die MOS- zu TTL-Pegelumsetzung. Bei typischen MOS-Schaltungen, die den herkömmlichen, hohen Spannungspegel am Ausgang aufweisen, kann ein einfacher Spannungsteiler verwendet werden, der ein Standard-LPS-TTL-Gatter speist. Der Spannungsteiler besteht gewöhnlich aus einem 10-kΩ-Widerstand zusammen mit einem 30-kΩ-Widerstand, der an −12 V angeschlossen ist. Der Teilerausgang wird mit dem Eingang eines 74LS00-Standard-LPS-TTL-Gatters verbunden.

Typische TTL- zu MOS-Treiberschaltung
(Bild 7.11)

Eine typische Schaltung, in der der TTL-Ausgang der Eingang der MOS-Schaltung ist, zeigt Bild 7.11. Der Ausgang des TTL-Gatters ist typisch gleich oder größer als +2,0 V bei hohem Logikpegel und +0,4 V bei niedrigem Logikpegel. Damit ein zuverlässiger Datentransfer gewährleistet ist, bedarf die Gate-Source-Spannung U_{GS} einer kritischen Betrachtung. Wenn U_{GS} größer ist als die Schwellspannung U_T (threshold voltage), wird T1 leitend. Aber wenn U_{GS} kleiner ist als U_T, sperrt T1.

Übertragungskennlinie eines TTL-Treibers
(Bild 7.12)

Die Übertragungskurven eines typischen TTL-zu MOS-Treibers sind in Bild 7.12 dargestellt. Bei geeigneter Wahl von U_{SS} weist der Ausgang des TTL-Gatters die richtige Polarität für die Gate-Source-Spannung auf, die zugleich die Eingangsspannung des MOS-Transistors ist. Bei einem P-Kanal-Bauelement sollte der Source-Anschluß mindestens auf dem Minimum der TTL-Versorgungsspannung von U_{CC} = 4,5 V, und der Drain-Anschluß auf Masse liegen. Damit werden die Gate-Source-Spannungen negativ, wodurch sie einen P-Kanal-Anreicherungstyp einschalten können.

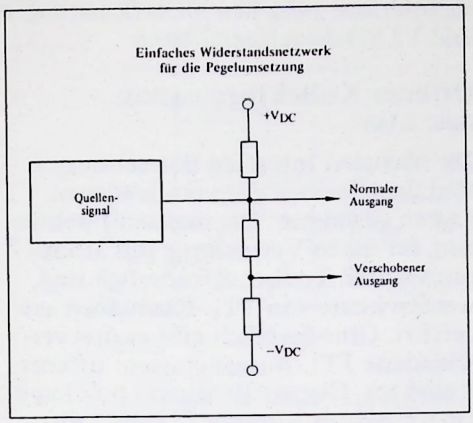


Bild 7.9
Einfaches Widerstandsnetzwerk für die Pegelumsetzung

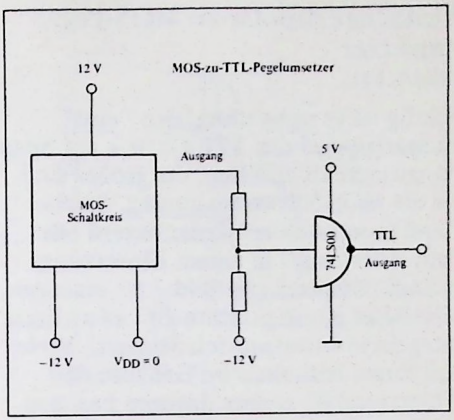


Bild 7.10
MOS zu TTL Pegelumsetzer

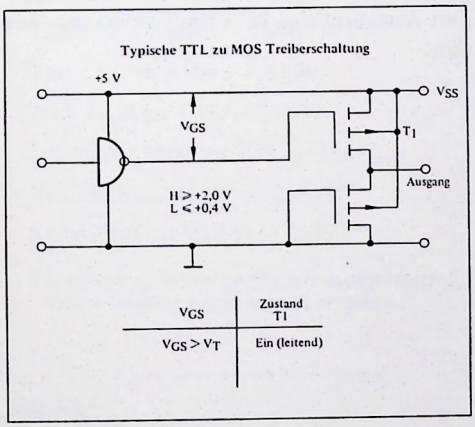


Bild 7.11
Typische TTL zu MOS Treiberschaltung

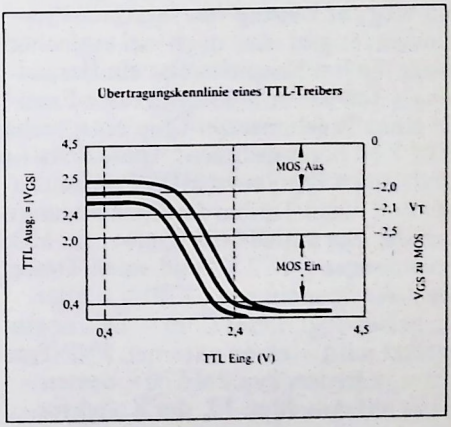


Bild 7.12
Übertragungskennlinie eines TTL-Treibers

Einfacher Bipolar zu MOS-Pegelumsetzer
(Bild 7.13)

Häufig ist es notwendig, den "eins" Ausgangspegel des TTL-Gatters auf Spannungspegel zu erhöhen, die größer sind als die MOS-Schwellspannung. Solche Pegel lassen sich entweder extern oder "auf dem Chip" in einem Umsetzungsprozeß entsprechend Bild 7.13 erzeugen. Mit dieser Konfiguration gibt es mehrere integrierte Interface-Schaltungen. Viele von ihnen enthalten im Gehäuse den Widerstand R; andere dagegen besitzen Treiberausgänge mit offenen Kollektoren und benötigen deshalb einen externen Widerstand.

Pegelumsetzer-Chip
(Bild 7.14)

Der Aufbau spezieller Schaltungen ist ein Weg zur Lösung von Interface-Problemen; es gibt aber noch viel einfachere Wege. So hat beispielsweise ein Hersteller als Teil seiner Mikroprozessor-Familie einen Pegelumsetzer-Chip entsprechend Bild 7.14 herausgebracht. Diese Schaltung stellt einen Bipolar-zu MOS-Pegelumsetzer und einen Treiber für hohe Spannungen dar. Sie enthält vier Treiber für MOS-Spannungen ($\leqslant$ 12 V) und einen Treiber für hohe Spannungen ($\geqslant$ 30 V). Allerdings benötigt dieser Chip – falls er eingesetzt wird – einen externen PNP-Transistor, z.B. den Typ 2N 5707, dessen Basis mit Anschluß 12, der Kollektor mit Anschluß 11 und der Emitter mit U_{DD} verbunden wird.

Typische Pegelumsetzer-Bausteine
(Bild 7.15)

Die kurze tabellarische Übersicht in Bild 7.15 zeigt die Vielfalt der erhältlichen Interface-Bausteine. Es handelt sich dabei um spezielle Bausteine, die die Pegelumsetzung von TTL auf MOS und CMOS sowie von MOS auf TTL vornehmen. Darüber hinaus gibt es spezielle Bausteine als Interface zwischen MOS-Schaltungen und VLED-Anzeigeeinheiten.

Offener Kollektorausgang
(Bild 7.16)

Die nächsten Interface-Betrachtungen sind dem Treiben der verschiedenen Lasten gewidmet. Die meisten Funktionen, für die in Verbindung mit Mikroprozessoren Treiber erforderlich sind, werden heute von TTL-Bausteinen ausgeführt. Grundsätzlich gibt es drei verschiedene TTL-Ausgangsarten: offener Kollektor, Gegentakt (totem pole) und Drei-Zustands-Ausgang (3-state). Der offene Kollektorausgang ist in Bild 7.16 dargestellt. Liegt bei dieser Schaltung der Ausgang auf null, so ist der Transistor T2 durchgeschaltet und sein Ausgang liegt auf Masse. Bei einer hohen Ausgangsspannung muß ein sog. Pull-up (Hochzieh-) Widerstand von außen an den Ausgang des IC's angeschlossen werden.

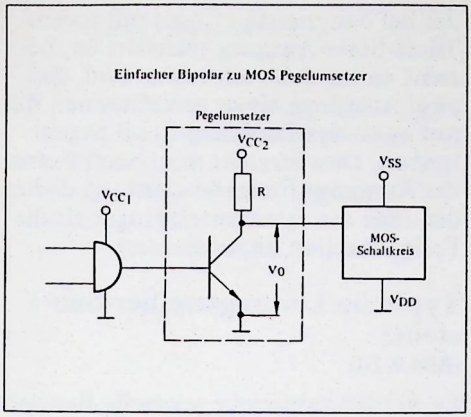


Bild 7.13
Einfacher Bipolar zu MOS Pegelumsetzer

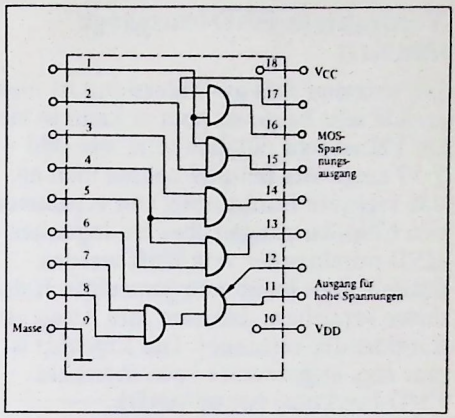


Bild 7.14
Pegelumsetzer-Chip

Vier TTL zu MOS – 75365

Vier TTL zu CMOS – 75367

Zwei MOS zu TTL – 75207

7-Segment MOS zu TTL – 75270

Vier MOS zu VLED – 75491*

Sechs MOS zu VLED – 75492*

* können zusammen für die Ansteuerung eines Displays verwendet werden.

Bild 7.15
Typische Pegelumsetzer-Bausteine

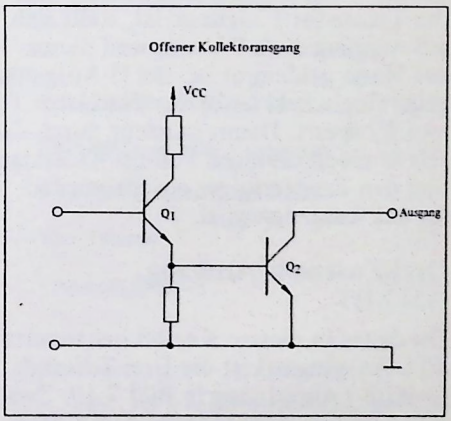


Bild 7.16
Offener Kollektorausgang

Verdrahtete UND-Ausgänge
(Bild 7.17)

Ein externer Pull-up-Widerstand ist nicht gerade sehr bequem; aber er kann in vielen Fällen sehr nützlich sein, wie Bild 7.17 zeigt. Als Beispiel nehme man an, daß mehrere Funktionen von verschiedenen Chip-Ausgängen über ein logisches UND miteinander verknüpft werden müssen. Dies kann man ganz einfach dadurch erreichen, daß man ihre Ausgänge miteinander verbindet. Das Ergebnis ist eine sog. angedeutete bzw. indirekte UND-Funktion (wired AND).

Gegentakt-Ausgang
(Bild 7.18)

Die nächste der erwähnten Treiberschaltungen ist der Gegentakt (totem pole) – Treiber entsprechend Bild 7.18. Wenn der Transistor T2 leitend ist, stellt sich am Ausgang etwa 0 V ein, weil dieser auf Masse geklemmt ist. Bei H-Ausgangspegel (logisch 1) leitet der Transistor T1 und T2 sperrt. Damit entsteht durch den relativ niederohmigen Pull-up-Widerstand und den Emitterfolger ein Strompfad für das Ausgangssignal.

Drei-Zustands-Ausgang
(Bild 7.19)

Die dritte in diesem Kapitel behandelte TTL-Ausgangsart ist die Drei-Zustands (3-state-) Anordnung in Bild 7.19. Zwischen diesem Ausgang und dem Gegentakt-Ausgang bestehen zwei grundsätzliche Unterschiede. Zusätzlich zu dem HIGH- (T1 leitend; T2 sperrt) und dem LOW- (T2 leitend; T1 sperrt) Zustand liefert er einen dritten hochohmigen Zustand (beide Transistoren T1 und T2 sperren). Diese Fähigkeit ist ein großer Vorteil bei Bus-Betrieb mit Mehrfach-Lasten, der sich besonders bei Mikroprozessoren bemerkbar macht. Der typische Sperrstrom einer solchen Schaltungsanordnung beträgt nur 40 μA im hochohmigen Zustand. Der zweite Unterschied, der bei den meisten Chips mit einem Three-State-Ausgang realisiert ist, besteht darin, daß verhindert wird, daß zwei Ausgänge einen gemeinsamen Bus mit zwei verschiedenen Logikpegeln treiben. Dies erreicht man beim Entwurf der Ausgangs-Freigabeschaltung dadurch, daß man die Sperrzeiten länger als die Freigabezeiten dimensioniert.

Typische Leitungstreiber-Bausteine
(Bild 7.20)

Es werden zahlreiche spezielle Bausteine angeboten, die als Leitungstreiber für ein Mikroprozessor-System entworfen worden sind. In der Tabelle in Bild 7.20 sind einige Typen aufgeführt, die diese Aufgabe erfüllen.

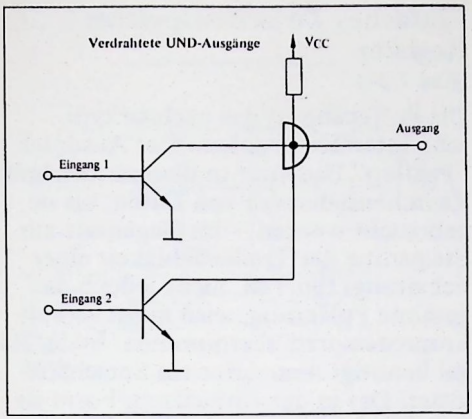


Bild 7.17
Verdrahtete UND-Ausgänge

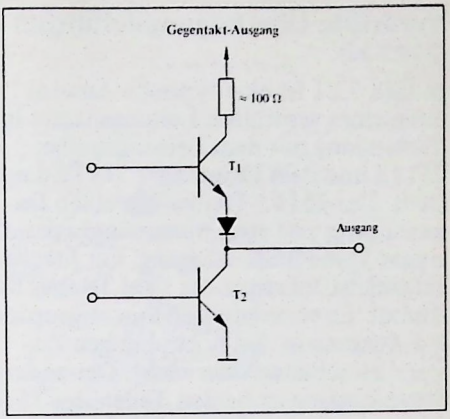


Bild 7.18
Gegentakt-Ausgang

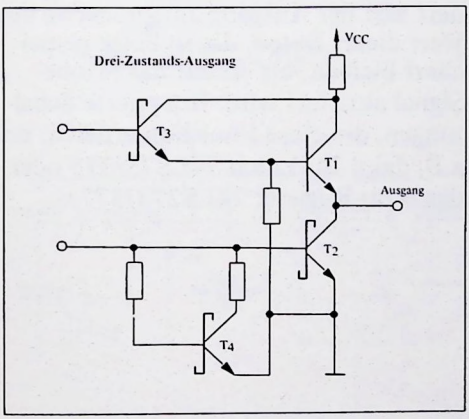


Bild 7.19
Drei-Zustands-Ausgang

Zwei differenzielle Treiber	75113
Zwei differenzielle Empfänger	75115
Vier Bus-Treiber/Empfänger	75138
Vier Treiber	75188
Vier Empfänger	75154

Bild 7.20
Typische Leitungstreiber-Bausteine

Verdrillte Übertragungsleitungen
(Bild 7.21)

In Bild 7.21 ist eine typische Anwendung eines verdrillten Leitungspaares in Verbindung mit dem Leitungstreiber 75113 und dem Empfänger 75115 dargestellt. Der 75113-Treiber hat einen Dateneingang und zwei Steuereingänge mit einem Three-State-Ausgang. Ein Steuereingang ist für einen der zwei Treiber bestimmt. Er veranlaßt den ihm zugeordneten Ausgang in den hochohmigen Zustand zu schalten oder nicht. Der andere Steuereingang ist beiden Teilen des 75113 zugeordnet, so daß er auch beide Teile in den hochohmigen Zustand schalten kann.

Typischer Party-Line-Bus
(Bild 7.22)

Eine andere Anwendung für den 75113- und 75115-Baustein ist der in Bild 7.22 gezeigte Party-Line-Bus und umfaßt fünf verschiedene Sende- und Empfangsstationen. Station 3 beinhaltet sowohl einen Sender als auch einen Empfänger. Falls diese Kombination an einem Ort erforderlich ist, können diese beiden durch einen einzigen Baustein, dem Sender/Empfänger (Transceiver) 75138, ersetzt werden.

Punkt-zu-Punkt-Anwendung des Transceivers 75138
(Bild 7.23)

Bild 7.23 zeigt ein Anwendungsbeispiel ähnlich dem Treiber/Empfänger in Bild 7.21. Allerdings wird in diesem Fall nur ein einziger Baustein benötigt. Eine typische Anwendung wäre hierfür beispielsweise eine 5-MHz-Datenübertragung zwischen zwei etwa 15 m entfernten Einheiten. Eine längere Leitung würde die Übertragungsfrequenz herabsetzen.

Einfaches Zwischenspeicher-Register
(Bild 7.24)

Die Pufferung ist das nächste typische Interface-Problem. Der Ausdruck "Puffern" bedeutet in diesem Abschnitt Zwischenspeichern von Daten, bis sie gebraucht werden – im Gegensatz zur Steigerung der Treiberfähigkeit einer Schaltung. Ein Teil, nicht jedoch die gesamte Pufferung, wird meist von Mikroprozessoren übernommen. In der Regel benötigt man dafür ein Speicherregister. Das in der einfachsten Form als sog. Latch entsprechend Bild 7.24 ausgeführt ist.

Wenn Daten am Eingang stabil anliegen, wird das Strobe-Signal aktiviert, und die Daten werden in das Latch eingespeichert. Während der Einspeicherung ändert sich der Ausgang entsprechend dem Wert dieser Daten, die so lange gespeichert bleiben, bis erneut das Strobe-Signal aktiviert wird. Integrierte Schaltungen, die diese Funktion erfüllen, sind z.B. das 4-Bit-Latch 74LS75/375 oder das 8-Bit-Register 74LS273/377.

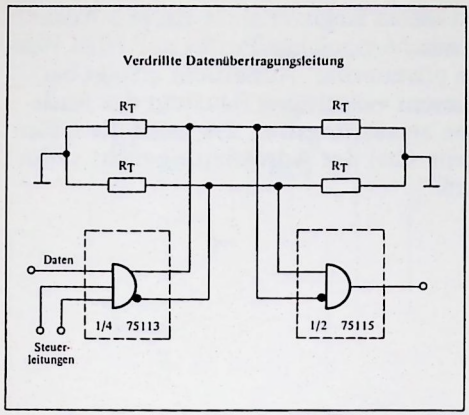


Bild 7.21
Verdrillte Datenübertragungsleitung

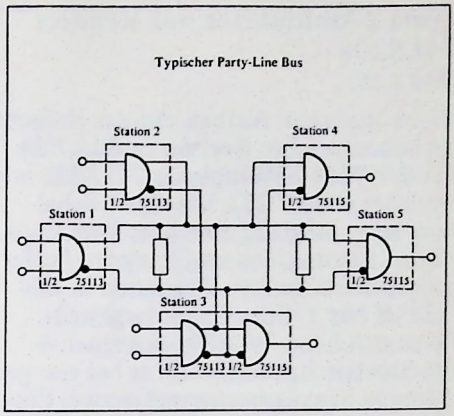


Bild 7.22
Typischer Party-Line Bus

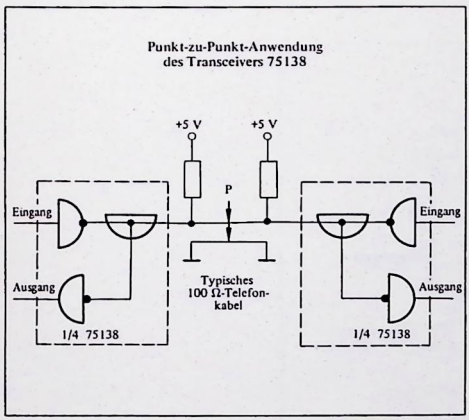


Bild 7.23
Punkt-zu-Punkt-Anwendung des Transceivers 75138

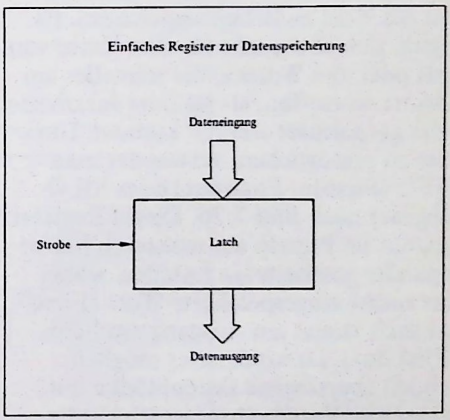


Bild 7.24
Einfaches Register zur Datenspeicherung

1-aus-2-Multiplexer mit Register 74LS298
(Bild 7.25)

Einen ähnlichen Aufbau wie das einfache Zwischenspeicher-Register in Bild 7.24 hat der 1-aus-2-Multiplexer 74LS298 mit Speicher (Bild 7.25). Mit dieser Schaltung ist es möglich, eine oder zwei Funktionen auszuwählen und in einem Puffer zu speichern. In dem Diagramm in Bild 7.25 ist nur 1/4 des Chips dargestellt. Da es sich beim 74LS298 um einen 4-Bit-Baustein handelt, kann er bei vier getrennten Signalgruppen mit je zwei Eingängen, einen Eingang auswählen und an den Ausgang schalten.

FIFO- oder SILO-Register
(Bild 7.26)

Das Latch in Bild 7.25 ist ausreichend um ein Wort zwischenzuspeichern. Es eignet sich aber nicht für eine Kette von drei oder vier Worten, die schneller am Zielort eintreffen, als sie dort verarbeitet oder gespeichert werden können. Um dies zu ermöglichen, verwendet man FIFO (First-in, First-out) oder SILO-Register nach Bild 7.26. Dieses Register besteht im Prinzip aus mehreren hintereinander geschalteten Latches, wobei das zuerst eingespeicherte Wort (First-in) auch zuerst am Ausgang erscheint (First-out). Dadurch ist es möglich, schnell übertragene Datenblöcke mit langsamer Verarbeitungsgeschwindigkeit zu verarbeiten.

54LS170/74LS170-Register
(Bild 7.27)

Der letzte zu besprechende Pufferbaustein ist ein kleines Zwischenspeicher-Register (scratch-pad register) entsprechend Bild 7.27. Dieses Bauelement ist organisiert in vier Worte zu je vier Bit. Da die Adressen der vier Worte auf dem Chip getrennt decodiert sind, können Daten zur gleichen Zeit eingeschrieben oder ausgelesen werden. Darüber hinaus ist dieses Register als N-Bit-pro-Wort-Zwischenspeicher/Puffer auf 1024 Worte erweiterbar. Außerdem erfolgt bei diesem vielseitigen Baustein das Auslesen zerstörungsfrei, das heißt die Daten gehen bei der Adressierung nicht verloren.

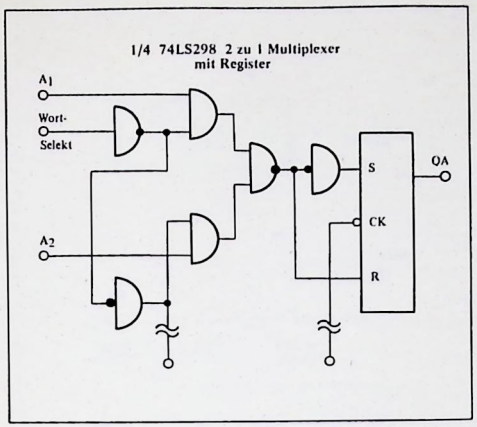


Bild 7.25
1/4 74LS298 2 zu 1 Multiplexer mit Register

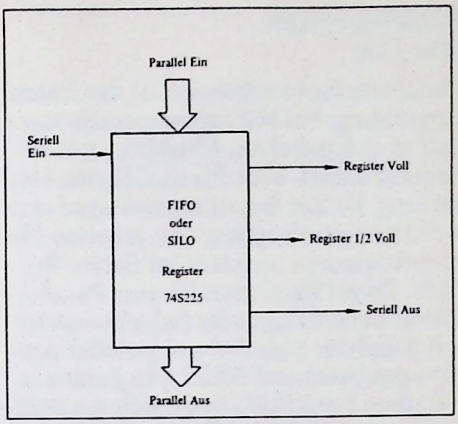


Bild 7.26
FIFO oder SILO Register

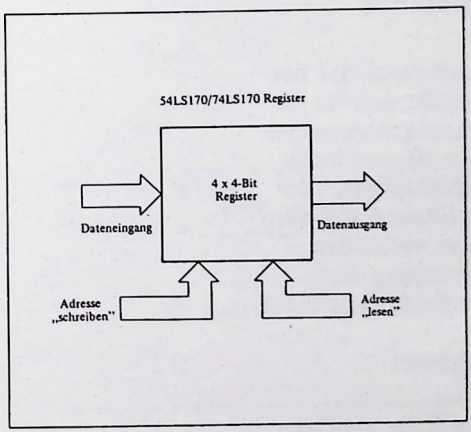


Bild 7.27
54LS170/74LS170 Register

Schieberegister
(Bild 7.28)

Der letzte Problembereich ist die Daten-Umsetzung. Bei Mikroprozessoren handelt es sich dabei am häufigsten um die Serien/Parallel- oder Parallel/Serien-Umsetzung. Da der Schaltungsaufwand notwendigerweise begrenzt ist, arbeiten Mikroprozessoren meistens im Serien-Betrieb. Damit kann man für eine Parallel/Serien-Umsetzung jedes Schieberegister mit paralleler Eingabe und serieller Ausgabe entsprechend Bild 7.28a benützen. Für diese Funktion eignen sich die Schieberegister 74LS95 (4 Bit) und 74LS165 (8 Bit). Die Serien/Parallel-Umsetzung ist in Bild 7.28b in vereinfachter Form dargestellt. Für diese Anwendung kann das 8-Bit-Schieberegister 74LS164 verwendet werden.

Schieberegister-Umsetzung
(Bild 7.29)

Jedes Schieberegister mit paralleler Ein- und paralleler Ausgabe läßt sich für die eben erwähnten Umsetzungen einsetzen (z.B. 74LS95). Hierzu muß man lediglich, wie das 4-Bit-Beispiel in Bild 7.29 zeigt, den Ausgang der Stufe A mit dem Eingang der Stufe B usw. verbinden. Der endgültige Serien-Ausgang wird durch den Ausgang der Stufe D gebildet.

Universal-Schieberegister
(Bild 7.30)

Um bei einer Anwendung die Zahl der unterschiedlichen Bausteine zu begrenzen, benützt man am besten ein Register, das alle in Bild 7.30 angegebenen Umsetzungen vornehmen kann. Dieses Bauelement besitzt parallele und serielle Ein- und Ausgänge sowie die Fähigkeit, nach links und nach rechts zu schieben. Ein 4-Bit-Beispiel ist der Baustein 74LS194 und ein 8-Bit-Beispiel der Baustein 74198.

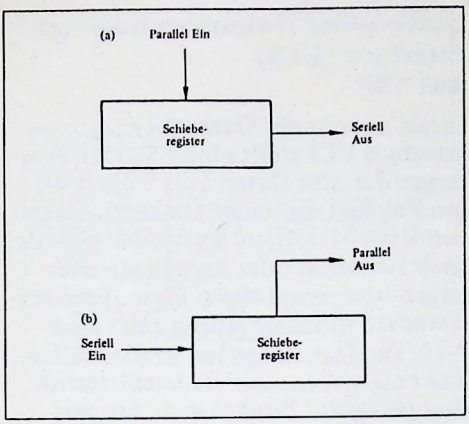


Bild 7.28
Schieberegister

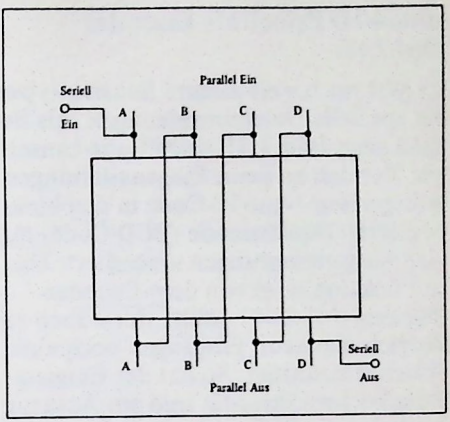


Bild 7.29
Schieberegister-Umsetzung

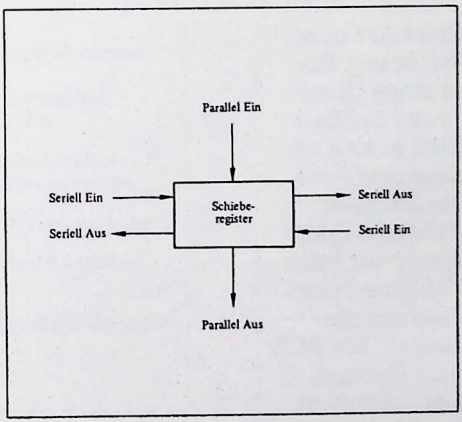


Bild 7.30
Universalschieberegister

4-aus-10 Prioritäts-Encoder
(Bild 7.31)

Es gibt noch viele andere Bausteintypen für spezielle Datenumsetzungen. Als Beispiel zeigt Bild 7.31 einen Code-Umsetzer, der den an neun Eingangsleitungen anliegenden 1-aus-10-Code in den binär codierten Dezimalcode (BCD-Code) mit vier Ausgangsleitungen umcodiert. Diese Funktion wird von dem Encoder-Baustein 74LS147 erfüllt, der jedoch zusätzlich den neun Eingängen noch eine Priorität zuordnet. So hat der Eingang 9 die höchste Priorität und am Ausgang erscheint die BCD-codierte Information "9", unabhängig davon, ob andere Eingänge (außer 9) ebenfalls logisch "Eins" sind oder nicht.

3-aus-8-Prioritäts-Encoder
(Bild 7.32)

Ein weiteres Beispiel dieser Art ist in Bild 7.32 dargestellt. Bei diesem Baustein handelt es sich um einen Oktal-Umsetzer, wobei die 1 aus 8 codierte 8-Bit-Eingangsinformation in eine oktalgewichtete 3-Bit-Ausgangsinformation umgesetzt wird. Der Baustein 74LS148 enthält ebenfalls eine Prioritätszurodnung der Eingänge wie beim 74LS147 (Eingang 7 ≙ höchste Priorität). Heute werden für nahezu alle Standard-Code-Umsetzungen, wie BCD zu Dual, Exzess-3-Gray zu Dezimal, BCD zu Dezimal, BCD zu 7-Segment usw., integrierte Schaltungen angeboten.

Typische spezielle Interface-Bausteine
(Bild 7.33)

In Bild 7.33 sind einige integrierte Bausteine aufgeführt, die eine komplexere Interface-Funktion erfüllen. Sie werden als Teil der neueren Mikroprozessor-Familien mehr und mehr auf dem Markt angeboten.

Universelles Datenübertragungs-Interface (UCI)
(Bild 7.34)

Dieses universelle Datenübertragungs-Interface UCI stellt einen Sender/Empfänger dar, der Daten von 5 bis 8 Bit pro Zeichen mit einer Geschwindigkeit von 0 bis 50 k Baud im Vollduplex-Betrieb synchron oder asynchron übertragen bzw. empfangen kann. Eine vereinfachte Blockschaltung zeigt Bild 7.34. Die Daten werden in Parallelformat eingegeben. Der Baustein verfügt über genügend Funktionen, um mit praktisch allen derzeitigen seriellen Datenübertragungstechniken zusammenzuarbeiten. Außerdem besitzt er interne Einrichtungen zur Erkennung von Paritäts-, Überlauf- und Formatfehlern.

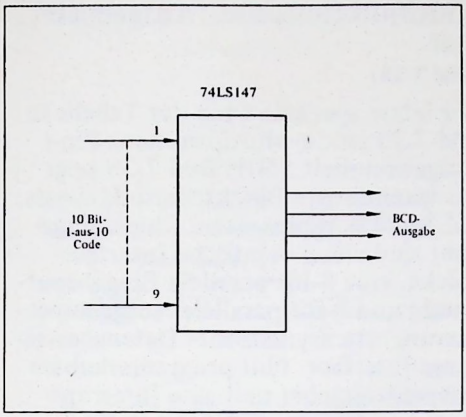


Bild 7.31
Ein 10-zu-4-Prioritäts-Encoder

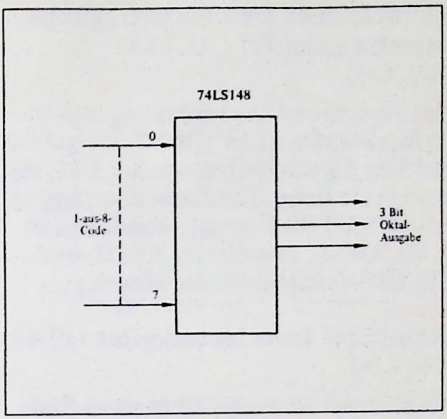


Bild 7.32
Ein 8-zu-3-Prioritäts-Encoder

Typische spezielle Interfacebausteine

8201	Universelles Datenübertragungs-Interface
6800 ACIA	Asynchroner Datenübertragungs-Interfaceadapter
6800 PIA	Peripherer Interfaceadapter
GP E/A	Universelle Ein-/Ausgabeadapter
5501	Multifunktions-Ein-/Ausgabeeinheit

Bild 7.33
Typische spezielle Interfacebausteine

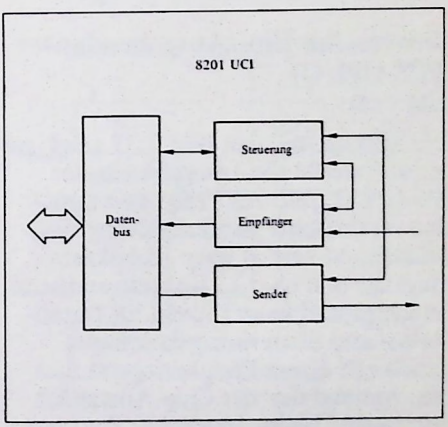


Bild 7.34
UCI (Universelles Datenübertragungs-Interface)

Asynchroner Datenübertragungs-Interfaceadapter (ACIA)
(Bild 7.35)

Der asynchrone Datenübertragungs-Interfaceadapter ACIA (Bild 7.35) hat die gleichen Eigenschaften wie der UCI, mit einer Ausnahme: Die Datenübertragung erfolgt im 9-Bit-Format. Außerdem ist er für ASCII-, erweitertes ASCII- und EBCDIC-Zeichenformate ausgelegt.

Peripherer Interfaceadapter (PIA)
(Bild 7.36)

Als nächster spezieller Chip ist in Bild 7.36 der periphere Interfaceadapter PIA dargestellt. Dieses bidirektionale Bauelement steuert die Daten- und Status-Information. Es besitzt zwei 8-Bit-Datenbusse, vier Unterbrechungsleitungen sowie eine Handshake-Steuerlogik.

Universeller Ein-/Ausgabeadapter (PPS-GPI/O)
(Bild 7.37)

Der Spezialchip, den Bild 7.37 zeigt, ist der universelle Ein-/Ausgabeadapter PPS-GPI/O. Bild 7.37 zeigt von ihm eine vereinfachte Blockschaltung. Das Bauelement verfügt über 12 diskrete Eingänge und über 12 diskrete statische Ausgänge und kann sowohl für Daten-, Status- und Steuerinformationsaustausch mit einem Peripheriegerät dienen. Anhand der vier Chip-Anwahlleitungen sind bis zu 16 solcher Bausteine direkt adressiert. Während des Betriebs bestimmt ein 4-Bit-Operationscode, ob oder ob nicht Daten über einen der zwei 4-Bit-Busse in die 4-Bit-Register A, B oder C der Eingabe oder der Ausgabe ein- oder ausgegeben werden. Der zweiphasige Eingangstakt wird auf dem Chip decodiert; gleichzeitig ist der A-Takt für externe Synchronisierzwecke über einen Treiber nach außen geführt.

Multifunktions-Ein-/Ausgabeeinheit
(Bild 7.38)

Der letzte spezielle Chip der Tabelle in Bild 7.33 ist die Multifunktions-Ein-/Ausgabeeinheit 5501. Bild 7.38 zeigt das vereinfachte Blockschaltbild dieses, z.Z. wohl komplexesten, Chips. Insgesamt sind vier wesentliche Interfaceblöcke, eine 8-Bit parallele Eingabepufferung und 8-Bit parallele Ausgabespeicherung, ein asynchrones Datenübertragungs-Interface, fünf programmierbare Intervallzeitgeber und eine Interruptsteuerung integriert. Über 4 Steuerleitungen können 12 verschiedene Operationen durchgeführt werden, wie z.B. laden eines Intervallzeitgebers mit einem Wert zwischen 64 μs und 16 ms oder laden des Baudratenregisters, welches 7 gängige Baudraten zwischen 110 Baud und 9600 Baud auswählt. Die Interruptsteuerung übernimmt die Steuerung des Interrupts zum 8080 Prozessor, hinsichtlich der zeitlichen Austaktung und Vektoradressengenerierung. Das Maskenregister des 5501 erlaubt dem Prozessor volle Kontrolle über alle Ein-/Ausgabefunktionen.

Zusammenfassung

Aus diesem Kapitel sollte hervorgehen, daß zahlreiche Bausteine für die verschiedenartigen Interface-Aufgaben zur Verfügung stehen. Es wurden alle Bereiche behandelt, um die Art der Probleme zu zeigen, die bei der Anwendung von Mikroprozessoren auftreten. Der Entwickler dürfte nun ein Gefühl für den Schaltungstyp bekommen haben, den er für eine bestimmte Funktion benötigt, sei es zur Pegelumsetzung, zum Treiben von Leitungen, zur Daten-Zwischenspeicherung/Pufferung oder zur Daten-Umsetzung.

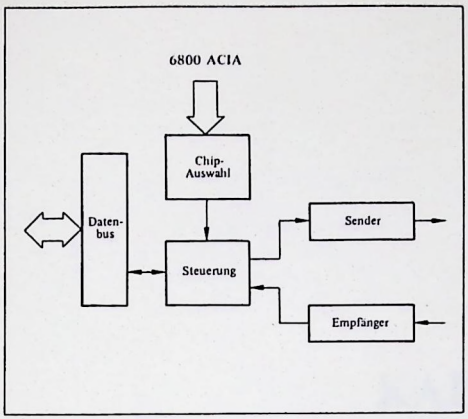


Bild 7.35
ACIA Asynchroner Datenübertragungs-Interfaceadapter

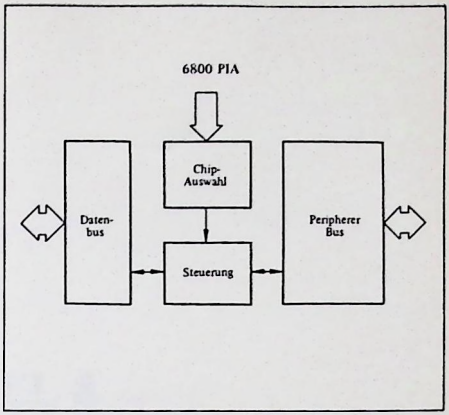


Bild 7.36
PIA Peripherer Interfaceadapter

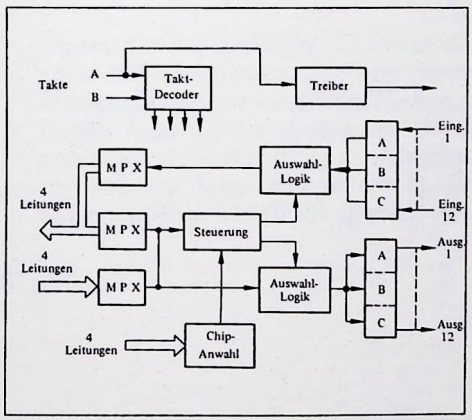


Bild 7.37
PPS-GP E/A Universeller Ein-/Ausgabeadapter

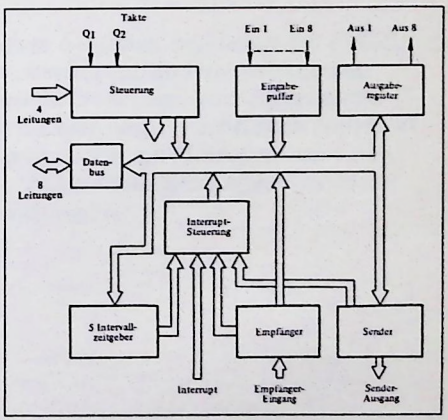


Bild 7.38
5501 Multifunktions-Ein-/Ausgabeeinheit

KAPITEL 8

POTENTIELLE ANWENDUNGEN UND GRENZEN

Dieses Kapitel definiert die unterschiedlichen Systeme, bei denen der Einsatz eines Mikroprozessors in Frage kommt. Weiterhin werden die wichtigsten Merkmale der zur Zeit erhältlichen und mit Mikroprozessoren aufgebauten Geräte besprochen. Außerdem werden dem Leser einige Richtlinien zur Hand gegeben, auf deren Basis er entscheiden kann, wann ein Mikroprozessor einzusetzen ist. Insgesamt führen die Erläuterungen in diesem Kapitel zu einem besseren Verständnis für speziellere Anwendungen.

Potentielle Anwendungen und Anwendungsgrenzen

In diesem Kapitel werden die Anwendungsgebiete und Anwendungsgrenzen der Mikroprozessoren geprüft. Es wird aufgezeigt, wie Mikroprozessoren die Kosten senken, die Entwicklungszeit des Produktes verkürzen und die Leistungsfähigkeit, Vielseitigkeit und Zuverlässigkeit desselben verbessern können.

Bei einigen der noch zu behandelnden Systeme stellt der Einsatz von Mikroprozessoren den Ersatz für Minicomputer oder festverdrahtete Logik dar. Diese Systeme schließen nachrichtentechnische Funktionen, Steuerungen, Terminals, Verkaufsstellen-Terminals und auch neue Produkte ein, die bisher ökonomisch nicht sinnvoll zu realisieren waren.

Gegenwärtige Mikroprozessor-Anwendungen
(Bild 8.1)

Bild 8.1 enthält eine Liste der Einsatzgebiete, auf die der Mikroprozessor bereits Einfluß genommen hat.

Klassifizierung der Anwendungen
(Bild 8.2)

Bild 8.2 gibt eine Einteilung der gängigen Anwendungsgebiete von Mikroprozessoren. Terminals und andere Peripheriegeräte sind naheliegende Anwendungen, da an sie immer höhere Anforderungen gestellt werden und neue Einsatzgebiete gefordert werden. Terminals können für Steuerungs- und arithmetische Aufgaben mit einem kleinen Speicher und etwas Logik ausgestattet werden. Sie können deshalb selbständig eine Datenvorverarbeitung durchführen.

Solche Terminals werden auch "intelligente Terminals" genannt.

Intelligente Verkaufsstellen-Terminals sind bisher mit standardisierten Logiknetzen festverdrahtet aufgebaut worden. Durch den Einsatz von Mikroprozessoren erhält man jedoch geringere Herstellungskosten, höhere Zuverlässigkeit und größere Vielseitigkeit, da sie mittels Software programmierbar sind. Die Verwendung von Mikroprozessoren in Terminals und allen anderen Peripheriegeräten kann in der Tat von wesentlichem Vorteil sein.

Hierarchisches Netzwerk
(Bild 8.3)

In großen hierarchischen Rechnernetzwerken kann der Mikroprozessor als Rechner auf der untersten Stufe eingesetzt werden (Bild 8.3). Eine Hauptaufgabe ist die Überwachung und Steuerung des Datenverkehrs innerhalb des Systems.

- Mikrocomputer
- Verteilte Computernetzwerke
- Automatische Typensetzmaschinen
- Inventurkontrolle
- Verkaufsstellen-Terminals
- Andere Terminals
- Nachrichtenübertragung
- Intelligente Instrumente
- Maschinensteuerung
- Adaptive Verkehrsregelung
- Ein-/Ausgabekanäle für Großrechner
- Medizinische Geräte
- Radionavigations-Geräte
- Optische Schriftzeichenerkennung (OCR)
- Automatisierte Tests
- Arbeitszeiterfassung
- Automobilsteuerungen
- Flugtechnische Geräte

Bild 8.1
Gegenwärtige Mikroprozessor-Anwendungen

- Terminals
- Mikrocomputer
- Nachrichtentechnik
- Steuerungen
- Konsumgüter

Bild 8.2
Klassifizierung der Anwendungen

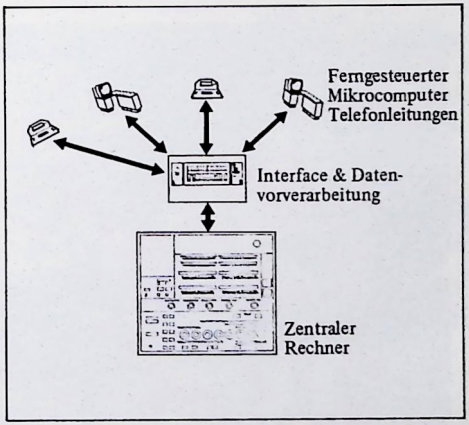


Bild 8.3
Hierarchisches Netzwerk

Das verteilte Computernetzwerk
(Bild 8.4)

Das Blockdiagramm eines verteilten Computers ist in Bild 8.4 dargestellt. Die Pipeline-Architektur und parallele Prozessoren werden bei einer Vielzahl von Maschinen zunehmend eingesetzt, nicht nur bei den Rechnergiganten. Eventuell werden solche Techniken auch in Mikroprozessorsystemen auftauchen.

Nachrichtenvermittlung
(Bild 8.5)

Die Nachrichtenübertragung erfordert die Durchführung von Aufgaben wie blockweise Datenübertragung, das Prüfen von Daten, das Codieren für die Übertragung, das Abfragen der Nachrichten auf spezielle Zeichenfolgen usw. Oft wurden für diese Funktion Minicomputer an jedem Ende der Nachrichtenverbindung verwendet, jedoch lassen die Bandbreite des Gesamtsystems und die gängigen Code-Formate den Einsatz von Mikroprozessoren jederzeit zu. Bild 8.5 zeigt den Einsatz von Mikroprozessoren in Relais-Stationen und in der Nachrichtenvermittlung bei Mikrowellen-Systemen.

Programmierbare Steuerungen und Regelungen
(Bild 8.6)

Eine programmierbare Steuereinheit ist in Bild 8.6 abgebildet. Sie eröffnet das ganze Gebiet der industriellen Prozeßsteuerung und -regelung, der Werkzeugmaschinen-Steuerungen, der Instrumentierungen, der Datenerfassungssysteme, des Schriftsetzens usw. Wiederum dienen die Mikroprozessoren in diesen Systemen als Ersatz entweder für festverdrahtete Logikschaltungen oder für Minicomputer. In der Prozeßregelung und -steuerung werden Mikroprozessoren an ein großes Spektrum von Meßfühlern angeschlossen. Sie fällen logische Entscheidungen, senden Steuersignale zu Motoren, Ventilen, Schaltern usw. und führen außerdem arithmetische Operationen wie z.B. Linearisierung, Skalierung und die Transformation von technischen Einheiten durch.

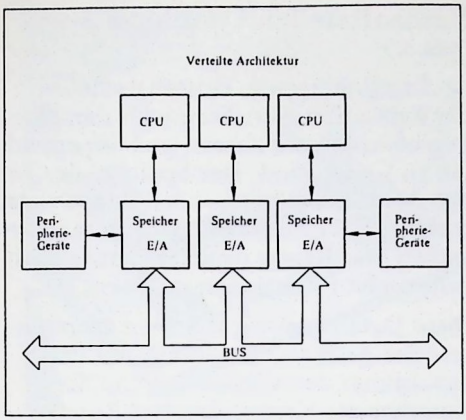


Bild 8.4
Das verteilte Computernetzwerk

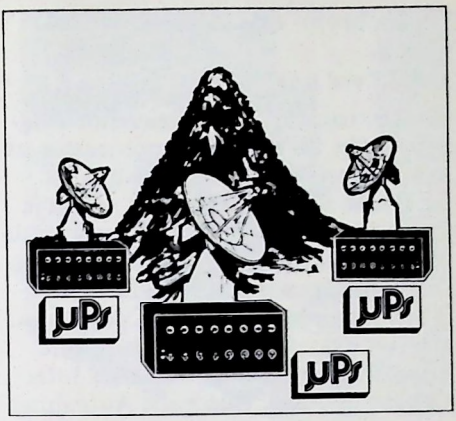


Bild 8.5
Nachrichtenvermittlung (μP = Mikroprozessor)

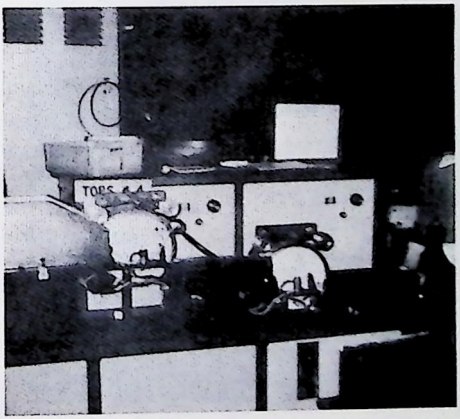


Bild 8.6
Programmierbare Steuereinheit

Fernschreib- und Bildschirm-Terminals
(Bild 8.7 und 8.8)

Eine der leicht zu identifizierenden Anwendungen für einen Mikroprozessor ist eine Datenstation, auch Terminal genannt. Die Bilder 8.7 und 8.8 zeigen je ein Standardexemplar eines Fernschreib- und Bildschirm-Terminals. Beide ermöglichen die Mensch-Maschine-Kommunikation für einen Rechner auf der Grundlage von sequentiellen Zeichenfolgen. Jedes einzelne Zeichen wird über Interrupt abgearbeitet. Durch die Aufnahme eines Speichers und etwas Steuerlogik in das Terminal kann die Informationsübertragung in größeren Blöcken erfolgen. Es kann ein Wort, eine Zeile oder sogar eine ganze Seite von 1600 bis 2000 Zeichen übertragen werden. Dabei kann ein Mikroprozessor als Steuerung für das Terminal dienen. Er gestattet dem Benutzer verschiedene, in lokalen Speichern abgelegte "Textseiten" zum Zwecke der Änderung oder der Ausgabe aufzurufen. Nach der Eingabe einer ganzen Seite, dem Einfügen spezieller Codes oder dem Tabellieren usw. kann der Benutzer eine Seite an die Zentraleinheit übertragen und eine andere Seite in das Terminal abrufen.

Die durchschnittliche Bildwechselfrequenz für einen Bildschirm beträgt 60 Bilder pro Sekunde. Bei einem Bildschirm mit 2000 Zeichen liegt die übliche Arbeitsweise bei einer Zeichenübertragungsdauer von 1/(2000 x 60) s, d.h. bei etwa 8 Mikrosekunden pro Zeichen. Deshalb würde eine solche Anwendung einen Mikroprozessor mit guten Ein/Ausgabemöglichkeiten, Interruptmöglichkeiten und mit Befehlsausführungszeiten für Datenübertragung im Bereich weniger Mikrosekunden erfordern. Zusätzlich müßte der Mikroprozessor mindestens 2 K x 8-Bit-Schreib/Lesespeicher (RAM) adressieren können.

Verkaufsstellen-Terminals
(Bild 8.9)

Andere intelligente Terminals stellen die Verkaufsstellen-Terminals dar, die in großen Warenhäusern und Supermärkten zu finden sind. Der Speicher des Terminals muß Verkaufs- oder Umsatzdaten enthalten, Korrekturen oder Zusätze erlauben und Belege drucken, sowie buchhalterische Funktionen zulassen.

Diese Daten können aus der Lagernummer des Artikels, dem Preis, der Personalnummer des Verkäufers und der Nummer der Kreditkarte des Kunden bestehen. Wenn alle Einzelheiten eines bestimmten Verkaufsvorganges eingegeben worden sind, berechnet das Terminal die Summe, die Umsatz- oder Mehrwertsteuer, die Gesamtsumme, druckt einen Beleg und überträgt vielleicht die Daten zu dem für die Inventarisierung und Bilanzierung verantwortlichen zentralen Rechner. Weil die meisten Funktionen mit durch den Menschen vorgegebenen Geschwindigkeiten durchgeführt werden und die Arithmetik einfach ist, werden höchstens etwa 25 Operationen pro Sekunde verlangt. Die Ausgabe über einen Drucker oder einen Bildschirm wird in der Regel mit 100 Zeichen pro Sekunde oder weniger bewerkstelligt.

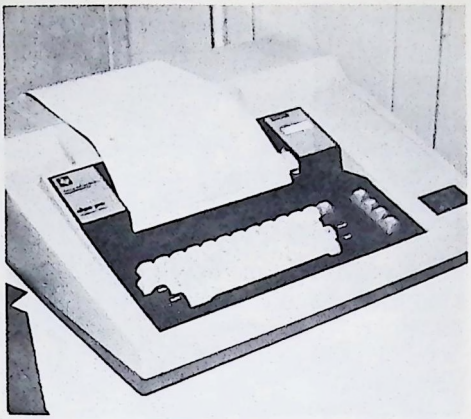


Bild 8.7
Fernschreib-Terminal

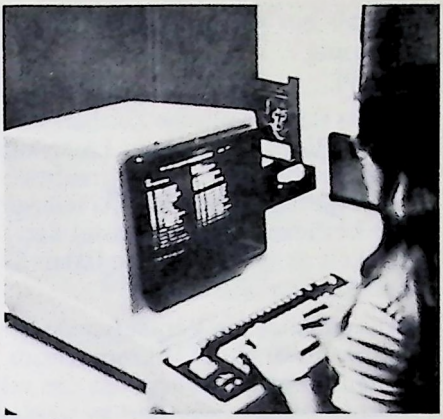


Bild 8.8
Bildschirm-Terminal

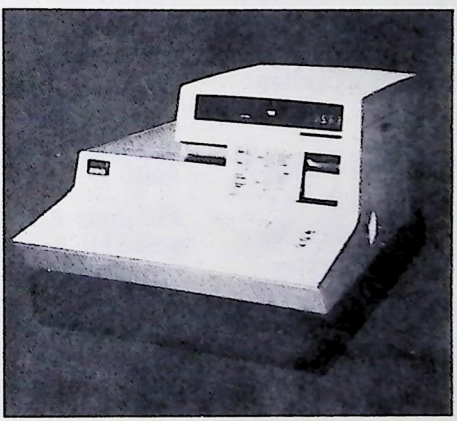


Bild 8.9
Verkaufsstellen-Terminal

Lesestab zur optischen Zeichenerkennung
(Bild 8.10)

Lesestäbe für die optische Zeichenerkennung (Bild 8.10) sind zur Entlastung des Verkaufspersonals von zeitraubenden und oft fehlerträchtigen Arbeitsvorgängen bestimmt. Der Stab tastet das Preisetikett ab und sendet die Daten direkt an das Terminal.

Bei Anwendungen in Verkaufsstellen-Terminals benötigt der Mikroprozessor mehrere Ein/Ausgabe-Ports, um den verschiedenen Aufgaben gerecht zu werden. Die Geschwindigkeit stellt kein besonderes Problem dar. Jedoch kann die Programmierung einer solchen Anwendung kompliziert werden, wenn Funktionen wie die Arbeitszeitregistrierung, Führen der Gehaltsabrechnung und der Bestandslisten enthalten sein sollen. Darum sollte man einen vielseitigen Befehlsvorrat und vielleicht große Speichermöglichkeiten in Betracht ziehen.

Steuerung von Peripheriegeräten
(Bild 8.11)

Die Steuerung eines Zeilendruckers ist ein Beispiel für eine Peripheriegeräte-Steuerung. Bild 8.11 zeigt einen Trommeldrucker, der einen Mikroprozessor dazu verwendet, die Hämmer so zu steuern, daß der richtige Abdruck von den auf der rotierenden Trommel erhaben angebrachten Zeichen entsteht. Die Ausgabegeschwindigkeit beträgt 1800 Zeilen pro Minute. Der Zeichenvorrat umfaßt 64 Zeichen und die Länge der Zeile besteht aus 132 Zeichen bzw. Spalten. Die Trommel dreht sich jede 1/30 Sekunde einmal um ihre Achse. Der Steuerung steht genau diese Zeitspanne zur Verfügung, um den Plan für alle Hämmer aufzustellen, die auf das Papier schlagen. Jeder der 132 Hämmer schlägt nur einmal pro Umdrehung an.

Berechnungen zum Zeilendrucker
(Bild 8.12)

Der Mikroprozessor muß ein Abbild von 64 x 132 Bits für die Hämmer erzeugen. Er erledigt dies in 1/30 Sekunde. Daher setzt er jede 3,9 Mikrosekunden ein Bit. Die zugehörigen Berechnungen sind in Bild 8.12 dargestellt.

Die zweite Aufgabe besteht aus der Übergabe der Abbildungsdaten an die Hämmer. Nur während weniger als 1/60 von der 1/30 Sekunde kann dem Hammer gestattet werden anzuschlagen. Diese 1/1800 Sekunde ist der Zeitraum, der für den Wechsel von einem Zeichen auf das benachbarte zur Verfügung steht (555 μs). Aus Experimenten ist bekannt, daß der Hammer nur 70 μs lang das Papier berühren darf. In diesem Zeitintervall müssen 132 Datenbits an die Magnete der Hämmer übertragen werden.

Ein Mikroprozessor mit einer Wortlänge von 4 Bit muß alle 70 μs 33 Worte auslesen, so daß ein Speicherwort von 4 Bit in weniger als 2 μs übertragen werden muß.

Für diese Anwendung kann typischerweise ein paralleler 4-Bit-Ein/Ausgabebus mit einer Adressierfähigkeit von 8 K Speicherplätzen und eine Mikroprogrammsteuerung mittels eines auf dem Prozessorchip angeordneten ROMs in Betracht gezogen werden.

Bild 8.10
Lesestab zur optischen Zeichenerklärung

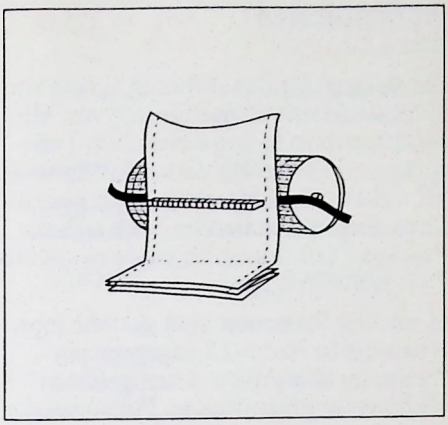


Bild 8.11
Steuerung von Peripheriegeräten

$$64 \times 132 = 8448 \text{ Bit}$$

$$\frac{1}{8448 \times 30} = \frac{1}{253{,}440}$$

$$= 0{,}0000039 \text{ s oder } 3{,}9\ \mu\text{s}$$

Bild 8.12
Berechnungen zum Zeilendrucker

Prüfautomaten
(Bild 8.13)

Ein Beispiel für den direkten Ersatz von Minicomputern ist der Einsatz von Mikrocomputern in automatischen Testsystemen. Testgeräte dieser Art können oft viel zweckmäßiger mit Mikroprozessoren ausgerüstet werden, weil sich die gleichen Funktionen für einen geringeren Preis realisieren lassen.

In solchen Systemen muß der Mikroprozessor die in Bild 8.13 angegebenen Parameter überprüfen. Dazu gehören die Spannungsversorgung, Pulsformen, Pulsfrequenzen und Datenmuster in der Eingabe. Gleichzeitig müssen die Datenmuster in der Ausgabe, Wellenformen am Ausgang und die Pegel geprüft werden. Solche Systeme werden für den Test von integrierten Schaltungen, Speichern und vielen anderen Bausteinen und Systemen gebaut.

Spezifikation von Mikroprozessoren für Prüfgeräte
(Bild 8.14)

Die Spezifikationen, die für einen Mikroprozessor nötig sind, der in Prüfgeräten eingesetzt werden soll, sind in Bild 8.14 aufgeführt.

Verkehrsregelung
(Bild 8.15)

Die Verkehrsregelung ist eine für Mikroprozessoren einfache Anwendung. Bild 8.15 zeigt eine typische Ausführung. Die bisher in solchen Systemen verwendeten Relais und die festverdrahtete Logik können durch ein für jede Kreuzung speziell entwickeltes (programmiertes) Mikroprozessorsystem ersetzt werden. Die Systemhardware ist standardisiert; nur das Programm im Festwertspeicher muß jeweils den spezifischen Gegebenheiten des Verkehrsflusses an einer Kreuzung angepaßt sein.

Für diese Anwendung ist ein langsamer 4-Bit-Mikroprozessor mit serieller Ein- und Ausgabe und einer Speicherkapazität von 256 Worten geeignet. Der Bedarf an Schreib/Lesespeichern kann mit den auf dem Mikroprozessorchip vorhandenen Registern gedeckt werden. Die Erfordernisse auf der Softwareseite werden ausreichend abgedeckt durch ein Simulationsprogramm und einen Assembler.

Speichergeschwindigkeit und Speichergröße
(Bild 8.16)

Jahrelang wurden im Rechnerentwurf die hierarchischen Speicherkonzepte verwendet. Wie aus Bild 8.16 hervorgeht, ist im allgemeinen ein Speicher um so langsamer, je größer er ist. Die Speicherhierarchie, besonders die von Cache-Speichern und virtuellen Speichern in Verbindung mit schnellen Speichern kleineren Umfangs, geben den Rechnern eine hinreichende Kapazität und Arbeitsgeschwindigkeit, um die Optimierung der Leistungsfähigkeit des Systems zu ermöglichen. Mit Mikroprozessoren ergeben sich ähnliche Möglichkeiten für die Zentraleinheit. Ein Teil der Rechenkapazität kann hier von Mikroprozessoren, die auf unterer Ebene in der Nähe der Peripheriegeräte arbeiten, wahrgenommen werden. Dadurch wird die enorm leistungsfähige Zentraleinheit frei für zeitraubendere Aufgaben, wie Matrizentransformation, schnelle Fouriertransformationen oder ähnliches.

Besonders im Entwurf verteilter (dezentralisierter) Rechner können Mikroprozessoren eingesetzt werden, die alle gleichzeitig arbeiten. Für eine solche Anwendung sind Pipeline- und Parallelprozessor-Architekturen notwendig. Deshalb sollte ein Mikroprozessor eine parallele Bus-Struktur besitzen, was bei neueren bipolaren Mikroprozessoren-Elementen der Fall ist, die aus kaskadierbaren 2- oder 4-Bit-Teilen (slices)

- Spannungsversorgung
- Pulsform
- Pulsfrequenz
- Eingabe-Datenmuster
- Ausgabe-Datenmuster
- Ausgangspulsformen
- Ausgangspegel

Bild 8.13
Prüfautomaten

Wortlänge	von mindestens 8 bis höchstens 16 Bit
Speicher	bis 32 K
Ein-/Ausgabe-kapazität	≈ für 32 Einheiten
Software	Cross-Assembler Editor Debugger

Bild 8.14
Spezifikationen von Mikroprozessoren für Testgeräte

Bild 8.15
Verkehrsregelung

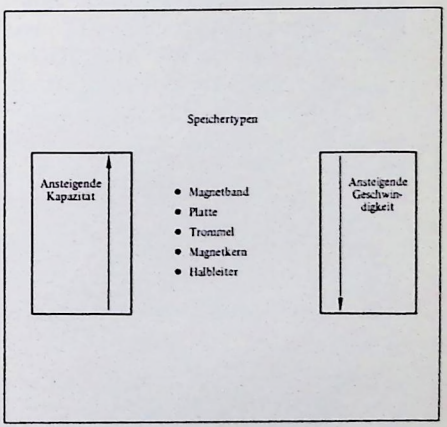


Bild 8.16
Speichergeschwindigkeit gegenüber Speicherkapazität

bestehen. Außerdem ist ein vielseitiges Bausteinsystem notwendig, das eine leichte Erweiterung oder Umgliederung des Rechnersystems gestattet. Ferner sollte in einer solchen Anwendung ein flexibles Softwaresystem mit guten Schnittstellen zu höheren Programmiersprachen vorhanden sein.

Rechnernetzwerk zur Verbrechensaufklärung
(Bild 8.17)

Bei der Informationsübertragung werden Mikroprozessoren für solche Aufgaben wie Vorverarbeitung der Identifikationszeichen und Codewandlung benutzt. Bild 8.17 zeigt ein großes, der Verbrechensaufklärung in den USA dienendes Rechnernetzwerk. Das Nachrichtenvermittlungssystem verarbeitet die Eingabe von 500 Terminals in einem Bundesland und verbindet die Landesbehörde mit den Bundesbehörden, um einen schnellen Zugriff auf die Vorstrafenverzeichnisse nehmen zu können. Durch Erledigen eines großen Teils an Datenvorverarbeitung in diesem Beispiel können Mikroprozessoren den Informationsaustausch von 1000 auf 10 000 Zeichen pro Sekunde erhöhen.

Übertragung von Differenzsignalen
(Bild 8.18)

Mikroprozessoren ermöglichen die Einführung eines Differenz-Algorithmus zur Codierung. Er verbessert die Leitungsausnutzung dadurch, daß er nur die Differenz zwischen zwei aufeinanderfolgenden abgetasteten Signalen statt den üblichen ganzen 8 Bit überträgt. Deshalb würden im Durchschnitt nur 3 oder 4 Bit übertragen, was die Effektivität der Übertragung mehr als verdoppelt. Für diese Anwendung sollte man einen 4- oder 8-Bit-Mikroprozessor auswählen, bei dem ebenfalls das Programmspeicher-ROM mit integriert ist und der eine begrenzte parallele und/oder serielle Ein/Ausgabekapazität besitzt.

Numerische Maschinensteuerungen
(Bild 8.19)

Eine numerische Werkzeugmaschinen-Steuerung ist in Bild 8.19 dargestellt. Der Mikroprozessor dient in diesem System zur Steuerung. Er verarbeitet Programme für die Auswahl des Bohrers, für die Positionierung, für den Vorschub und die Bohrtiefe. Diese Programme werden z.B. von einem Lochstreifen gelesen und in die entsprechenden Steuersignale umgesetzt, die ihrerseits die Stellglieder ansteuern. Das Ergebnis wird gemessen, der nächste Programmschritt ausgeführt.

Für diese Anwendung sollte eine gute parallele Buskapazität vorhanden sein, um eine leichte Anpassung an 10- bis 12-Bit-Eingänge für Analog/Digital-Umsetzer bzw. Ausgänge für Digital/Analog-Umsetzer zu ermöglichen. Außerdem ist eine hohe Ein/Ausgabe- und Interrupt-Reaktionsgeschwindigkeit notwendig. Diese Anwendung erfordert eine große RAM-Programmspeicherkapazität und einen nichtflüchtigen Speicher, z.B. ein ROM oder PROM, für ein Anlaufprogramm. Die Unterstützung durch komplexe Software einschließlich höherer Programmiersprachen ist außerdem wünschenswert.

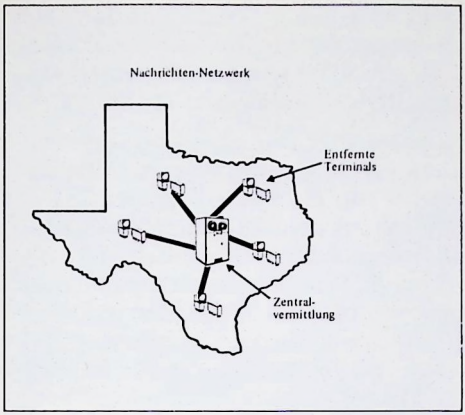


Bild 8.17
Informationsnetzwerk zur Verbrechensaufklärung

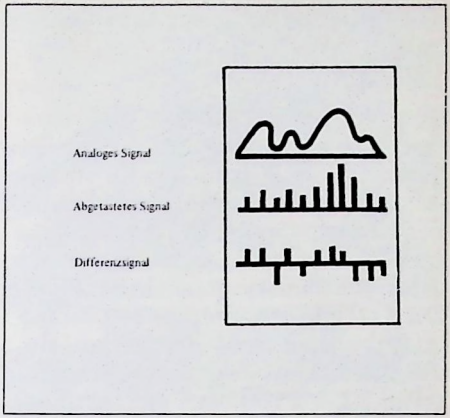
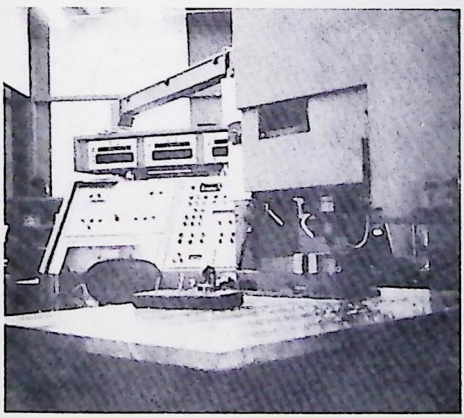


Bild 8.18
Differenzsignal-Übertragung



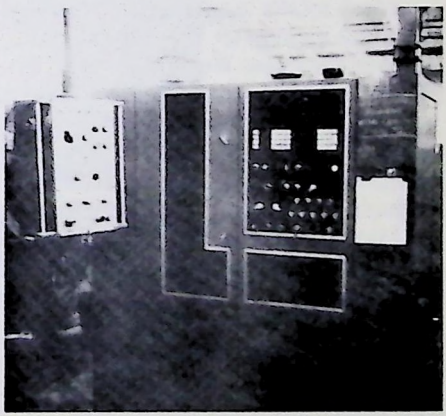


Bild 8.19
Numerische Werkzeugmaschinen-Steuerung

Anwendungen in der Kfz-Technik
(Bild 8.20)

Folgende Produkte in der Automobil-Industrie wären denkbar: Systeme zur Erhöhung des Wirkungsgrades und zur Abgaskontrolle, Antiblockiersysteme für die Bremsen und Diagnosesysteme im Fahrzeug. Ein Mikroprozessor ist fähig, alle diese Funktionen auszuüben. Bei Antiblockiersystemen besteht die Aufgabe darin, das Eingangssignal für die hydraulische Bremse sehr schnell (etwa 500 bis 1000 mal pro Sekunde) zu schalten, um die optimale Bremskraft zu erhalten.

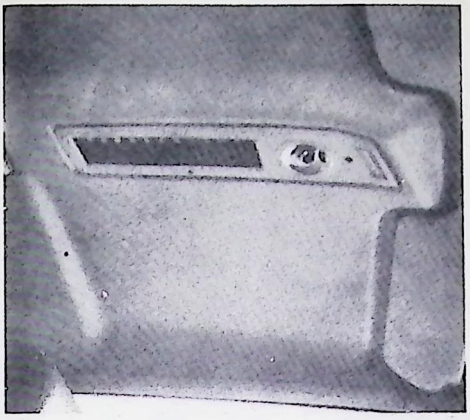
Hochentwickelte Displays (Bild 8.20 links), die den Benutzer über den Systemzustand informieren, können leicht mit einem Mikroprozessor realisiert werden. Diagnosesysteme für Automobile erfordern relativ langsame Mikroprozessoren mit begrenzter Ein/Ausgabefähigkeit und mit einer Wortlänge von 8 bis 16 Bit. Der Speicherbedarf an ROM und RAM ist begrenzt. In heutigen Systemen werden nur 256 Worte ROM und weniger als 128 Worte RAM benötigt. Deshalb könnte die Programmierung in der Maschinensprache erfolgen, eine Simulation des Programms ist jedoch wünschenswert.

Typische Anwendungen auf dem Unterhaltungssektor
(Bild 8.21)

Mikroprozessoranwendungen für Unterhaltungszwecke reichen von einer einfachen sequentiellen Steuerung mit serieller Ein/Ausgabe und einem ROM auf dem Prozessorchip bis zu einer sehr komplizierten Anwendung mit einer Vielzahl von Ein- und Ausgängen und mit einem Speicherumfang von 64 K.

Bild 8.21 zeigt eine solche Anwendung. Das Schachspielen erfordert ein für einen Mikrorechner sehr anspruchsvolles Programm und einen großen Speicher. Die einzelnen Züge beim Schachspielen und Teile der Strategie können durch steckbare ROMs implementiert werden.

Konsumgüter sind eine Hauptanwendung von Mikroprozessoren. Man betrachte nur die Bedeutung der LSI-Technologie auf dem Konsumgüter-Markt am Beispiel des elektronischen Taschenrechners. Manche Taschenrechner sind programmierbar; sie werden jedoch nicht als Mikroprozessoren bezeichnet. In Zukunft werden noch mehr Gebiete durch die Mikroprozessor-Technologie beeinflußt werden.



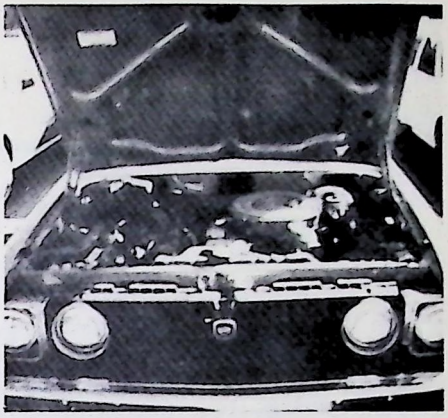


Bild 8.20
Anwendungen in der KFZ-Technik

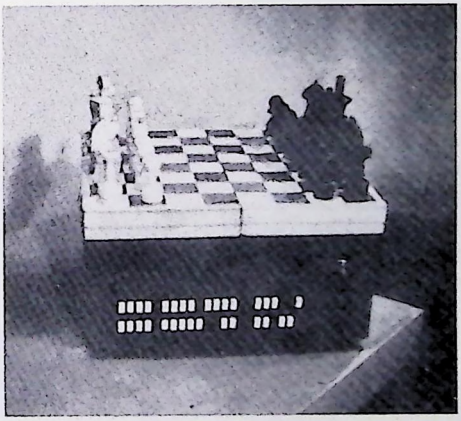


Bild 8.21
Typische Anwendung auf dem Unterhaltungssektor

Grenzen der heutigen Mikroprozessoren
(Bild 8.22)

Bild 8.22 führt einige der hauptsächlichen Grenzen der heutigen Mikroprozessoren auf. Jede Anwendung, bei der mehr als eine Wortlänge von 16 Bit benötigt wird, erfordert spezielle Methoden bei den meisten der heutigen Chips.

Andere Einschränkungen ergeben sich bei den Fähigkeiten zur Unterbrechungsbehandlung und der Zahl der Ein- und Ausgänge. Bei den heutigen Mikroprozessoren sollte ab etwa 16 Interrupts und 256 oder mehr Ein-/Ausgabe-Ports ein größeres System als einen Mikroprozessor in Betracht gezogen werden.

Die dritte Grenze bei Mikroprozessoren ist der Speicherumfang. Wegen der limitierten Bus- und Datenformatbreite des Mikroprozessors ist es nicht empfehlenswert, ein System mit mehr als 64 K Byte Speicher zu wählen.

Die Geschwindigkeit ist ein weiterer begrenzender Faktor. Die Befehlsausführungszeiten der Mikroprozessoren sind um eine Größenordnung langsamer als bei mit Schottky- oder ECL-Bausteinen aufgebauten Minicomputern. Deshalb muß man einen Minicomputer vorziehen, falls maximale Geschwindigkeit gefordert wird.

Zusammenfassung

In diesem Kapitel wurden mehrere Anwendungsgebiete untersucht und die für diese Anwendungen wichtigen Parameter der Mikroprozessoren bestimmt. Vier Grenzen wurden diskutiert, die bei den gegenwärtig erhältlichen Mikroprozessoren vorzufinden sind.

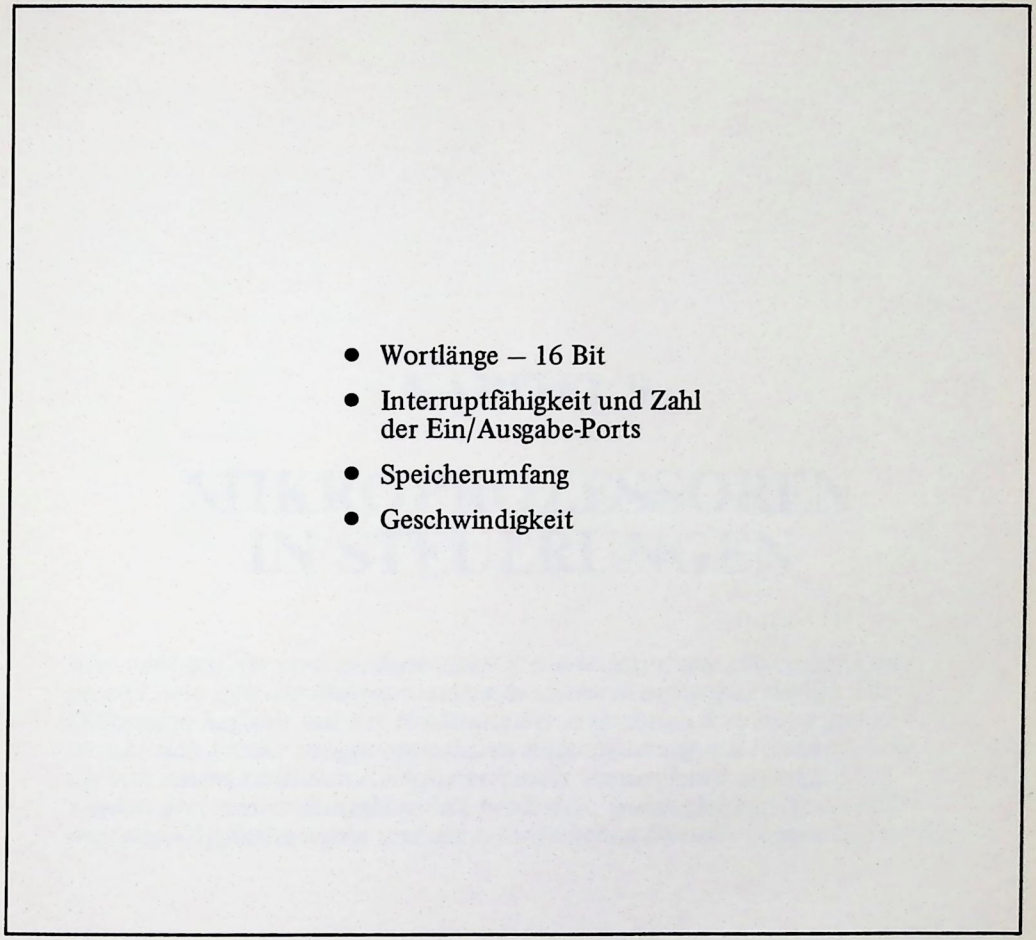


Bild 8.22
Grenzen der heutigen μProzessoren

KAPITEL 9

MIKROPROZESSOREN IN STEUERUNGEN

Hier wird auf die verschiedenartigen Steuerungssysteme eingegangen und gezeigt, wie sich der Mikroprozessor in solche Steuerungen einfügt. Die Diskussion beginnt mit der Erklärung der einfachsten Steuerung und erstreckt sich bis zur programmierbaren Folgesteuerung mit Rückkopplung, die von einem zentralen Rechner gesteuert werden kann. Anschließend werden drei Anwendungsbeispiele behandelt, wobei gleichzeitig die Mikroprozessor-Anforderungen und die erforderlichen Bauteile besprochen werden.

Mikroprozessoren in Steuerungen

Ein vielversprechendes und zugleich bedeutendes Gebiet für den Einsatz von Mikroprozessoren können in der Zukunft industrielle Steuerungen sein. Innerhalb dieses Kapitels werden die Erfordernisse dieses allgemeinen Anwendungsbereichs näher untersucht. Darüber hinaus wird gezeigt, wie Mikroprozessoren viele der bestehenden Anforderungen erfüllen und vielleicht auch neue Anwendungen stimulieren können.

Folge von Ereignissen
(Bild 9.1)

Grundsätzlich ist eine Steuerung ein Gerät, das eine Reihe von Ereignissen veranlaßt, die entsprechend Bild 9.1 in einer vorgegebenen Reihenfolge stattfinden. Diese Definition besagt, daß die Ereignisse zu geeigneten Zeitpunkten eintreten, da ein Ereignis möglicherweise eine fortlaufende oder auch Warte-Operation sein kann.

Steuerungssystem
(Bild 9.2)

Nach der klassischen Steuerungs- bzw. Regelungstheorie werden Systeme eingeteilt in Steuerungssysteme und Regelungssysteme. Die Struktur eines Steuerungssystems ist sehr einfach, wie Bild 9.2 zeigt. Da diese Steuerung keine Rückkopplung besitzt, kann sie nicht feststellen, ob das System richtig reagiert. Arbeitet das Steuerelement als Ein/Aus-Schalter oder digital, ist keine Rückkopplung notwendig, weil durch das Ein- und Ausschalten des Steuerelementes die gewünschte Systemantwort gegeben ist. Handelt es sich jedoch bei den Steuerelementen um analoge Bauteile, kann es gegebenenfalls wichtig sein zu wissen, ob der Systemausgang den richtigen Pegel erreicht hat und – falls nicht, wie groß der am Ausgang entstandene Fehler ist.

Regelungssystem
(Bild 9.3)

Eine Fehlerkorrektur läßt sich mit der in Bild 9.3 gezeigten Regelungsstruktur durchführen. Dem Regler werden über die Rückführsignale der Systemzustand und die Ausgangspegel mitgeteilt. Durch Vergleich mit den Eingangssignalen stellt er fest, ob die Systemantwort korrekt ist. Bei einer fehlerhaften Systemantwort werden die Regelsignale entsprechend geändert. Eine andere Bezeichnung für eine solche Struktur ist Rückkopplungssystem, wegen der vorhandenen Rückkopplungsschleife.

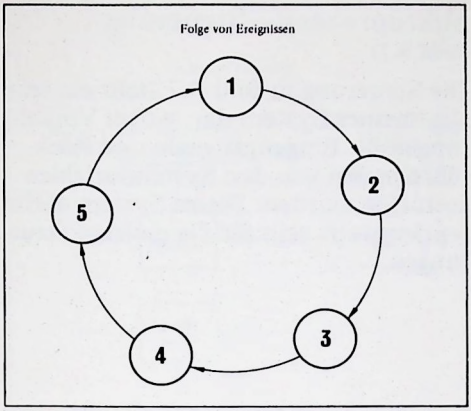


Bild 9.1
Folge von Ereignissen

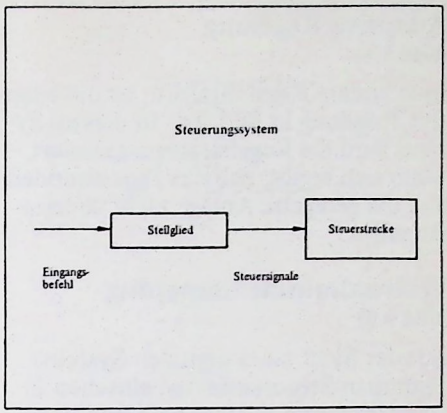


Bild 9.2
Steuerungssystem

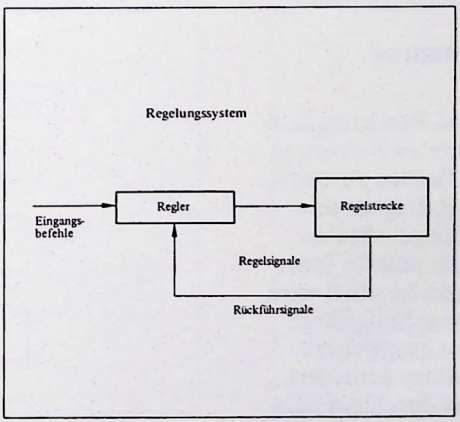


Bild 9.3
Regelungssystem

Adaptive Regelung
(Bild 9.4)

Eine andere Regelungsform ist die adaptive Regelung in Bild 9.4. In diesem System wird die Regelstrategie geändert, wenn sich ergibt, daß das Prozeßmodell und die geregelte Anlage nicht übereinstimmen.

Festverdrahtete Steuerung
(Bild 9.5)

Aus der Sicht eines digitalen Systems kann man Steuereinheiten einteilen in festverdrahtete Steuerungen und programmierte Steuerungen. Die festverdrahtete Steuerung nach Bild 9.5 stellt gewissermaßen eine "Folgemaschine" dar, die Steuersignale in einer bestimmten Reihenfolge erzeugt. Dieser Aufbau hat (im Bild gezeigt) einige Vorteile.

Programmierte Steuerung
(Bild 9.6)

Der hohe Entwurfs- und Fertigungsaufwand einer festverdrahteten Steuerung dürfte für komplexere Geräte als einfache Folgesteuerungen untragbar sein. Die Alternative zur festverdrahteten Steuerung ist die programmierte Steuerung nach Bild 9.6. In diesem Fall wird die erwünschte Folge von Ereignissen in einem Steuerspeicher gespeichert. Der Programmschrittzähler adressiert den Steuerspeicher und durchläuft diesen sequentiell. Jeder Steuerbefehl wird in einer Decodierschaltung interpretiert. Schließlich wird der Befehl durch die erzeugten Steuersignale ausgeführt. Die programmierte Steuerung ist vielseitig und einfach zu handhaben. Allerdings wird ihre Geschwindigkeit begrenzt durch die Speicher-Zugriffszeit, die Befehlsdecodierung und die Ausführungszeiten. Das besonders augenfällige Merkmal der programmierten Steuerung ist ihre Struktur, die im Grunde der eines Mikroprozessors mit einem Steuerspeicher entspricht.

Mikroprozessor-Steuerung
(Bild 9.7)

Die Steuerung in Bild 9.7 stellt ein sehr allgemeines System dar, wobei Vorkehrungen für Eingangssignale und Rückführsignalen von den Systemvariablen getroffen wurden. Dieses System dürfte repräsentativ sein für die meisten Steuerungen.

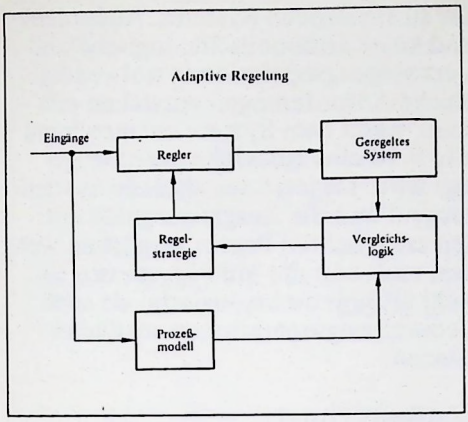


Bild 9.4
Adaptive Regelung

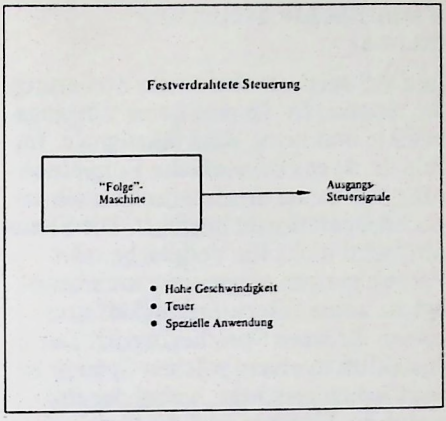


Bild 9.5
Festverdrahtete Steuerung

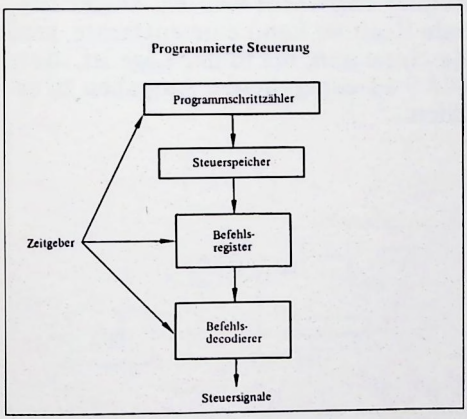


Bild 9.6
Programmierte Steuerung

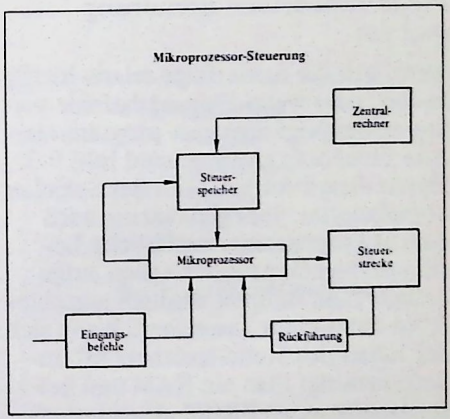


Bild 9.7
Mikroprozessor-Steuerung

Vereinfachte Steuerung
(Bild 9.8)

Bild 9.8 zeigt die einfachste Steuerung, die möglich ist. Es gibt keine Eingangsbefehle und keine Rückführsignale. Im Prinzip ist es eine einfache Folgesteuerung, die weder arithmetische noch logische Operationen ausführt. Die Steuerung wird nicht für Vergleiche oder Verzweigungen eingesetzt, auch benötigt sie keine Interruptfähigkeit und keinen direkten Speicherzugriff. Der Signalfluß in einem solchen System ist nach außen gerichtet, wobei der Prozessor die Flußrate und die Geschwindigkeit steuert. Wenn sich die Steuerfolge kaum ändert, kann der Steuerspeicher entweder ein ROM oder PROM sein.

Programmierbare Steuerung
(Bild 9.9)

Wenn sich die Steuerfolge relativ häufig ändert, oder wenn Eingangsbefehle vorliegen, benötigt man eine programmierbare Steuerung entsprechend Bild 9.9. Durch diesen Aufbau wird der Speicher komplizierter, aber gleichzeitig wird dem Mikroprozessor zusätzliche Leistungsfähigkeit gewissermaßen aufgedrängt. Zum Beispiel wird ein einzelnes ROM nicht mehr ausreichen. Wenn sich der Inhalt des Steuerspeichers oft ändert, benötigt man ein RAM und beispielsweise einen PROM als dauerhaften Speicher, in dem das momentane Steuerprogramm im Falle eines Stromausfalles zur Sicherheit abgespeichert wird.

Mikroprozessor-Regelung
(Bild 9.10)

Für programmierbare Steuerungen sind ein RAM und/oder PROM, Interruptfähigkeit und direkter Speicherzugriff erforderlich. Dennoch hat das Gerät in Bild 9.9 immer noch einen nach außen gerichteten Signalfluß, also in Richtung des zu steuernden Systems. Außerdem sind keine arithmetische, logische und Verzweigungsoperationen notwendig. Solche Anforderungen entstehen erst dann, wenn dem System entsprechend Bild 9.10 eine Rückführung hinzugefügt wird. Da jetzt der aktuelle Systemzustand und die Ausgangssignale mit den erwünschten Pegeln verglichen werden, muß nun der Mikroprozessor sowohl arithmetische, logische als auch Verzweigungsoperationen ausführen können.

Zentralrechner
(Bild 9.11)

Falls es notwendig sein sollte, einen kompletten Prozeß zu regeln, könnte die Programmierung der Eingangsbefehle und Steuerfolgen von einem Zentralrechner überwacht werden. Dieser zentrale Rechner kann eine entfernte, große Maschine sein, die in der Lage ist, die in Bild 9.11 angegebenen Aufgaben zu erfüllen.

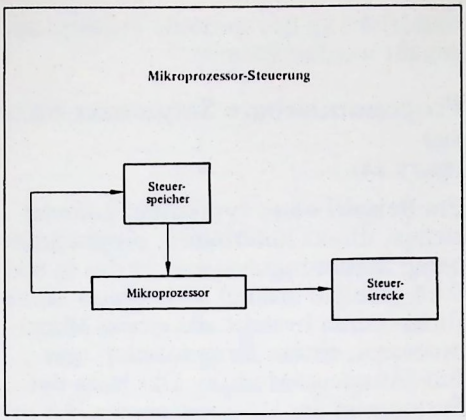


Bild 9.8
Vereinfachte Steuerung

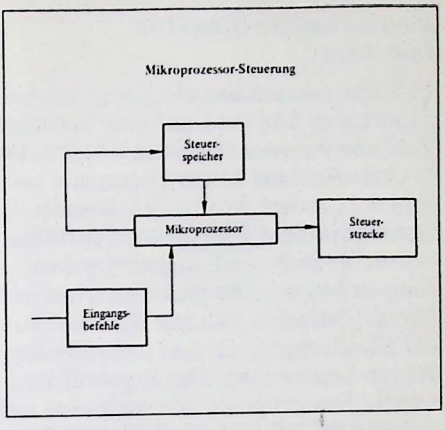


Bild 9.9
Programmierbare Steuerung

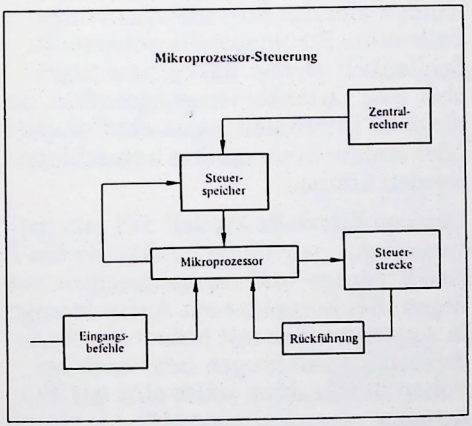


Bild 9.10
Mikroprozessor-Regelung

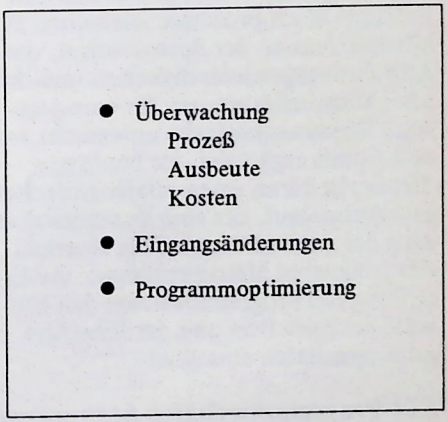


Bild 9.11
Aufgaben des Zentralrechners

Hierarchische Struktur
(Bild 9.12)

In Wirklichkeit kann die hierarchische Struktur in Bild 9.11 auf eine beliebige Zahl von Niveaus erweitert werden. Der höchste Rechner in der Hierarchie nach Bild 9.12 liefert Ausdrucke, Inventarlisten und andere Management-Informationen, ebenso wie Langzeit-Optimierungsstrategien. Die Maschinen unteren Niveaus befassen sich mit der Ausführung der Gleichungen, die den individuellen Prozeß beschreiben. Das Ergebnis sind sowohl Steuersignale wie auch eine individuelle Prozeß-Optimierung.

Das Steuerungs-Entwurfsproblem läßt sich auf die Auswahl der Mikroprozessor-Familie reduzieren und anschließend kann sofort das Steuerprogramm für das Problem geschrieben werden. Um den richtigen Mikroprozessor aussuchen zu können, müssen der Befehlsvorrat, die Ausführungsgeschwindigkeiten und die Ein-/Ausgabefähigkeiten für eine gegebene Steuerungsaufgabe untersucht werden. Somit ergibt sich die benötigte Steuerung durch einen relativ einfachen Entwurfsablauf, der eine Zusammenfassung der Anforderungen, die Auswahl des geeigneten Mikroprozessors, die Erstellung des Programms sowie den Entwurf des Speichers und der Ein-/Ausgabekapazitäten einschließt.

5TI-Programmierbares Steuerungssystem
(Bild 9.13)

Eine Möglichkeit für den Entwurf einer programmierbaren Steuerung besteht darin, das Entwurfsprogramm entsprechend Bild 9.13 mit einem Mikroprozessor zu implementieren. Falls jedoch das Steuerungsproblem völlig digitaler Art ist, also mit Ein/Aus-Zuständen für Ein- und Ausgangssignale, ist es sinnvoll, eine der kommerziell erhältlichen programmierbaren Mikroprozessor-Steuerungen (Sequenzer) auszuwählen, die schnell und leicht an das spezielle Problem angepaßt werden können.

Programmierbare Sequenzer-Struktur
(Bild 9.14)

Ein Beispiel eines typischen, kommerziellen, direkt lieferbaren, programmierbaren Steuerungssystems ist das in Bild 9.14 gezeigte Modell 5TI. Dieses Steuerungssystem besteht aus einem Mikroprozessor, einem Programmier- und Ein-/Ausgangs-Moduln. Das Herz des Systems ist der Mikroprozessor, der natürlich den Mikroprozessor selbst und den Speicher einschließt. Dieses Subsystem folgt den Befehlen, interpretiert sie und führt sie aus, indem es die geeigneten Ausgangssignale erzeugt. Außerdem ist ein PROM-Programmierer vorhanden, der das Steuerprogramm im Falle eines Stromausfalls sicherstellt. Schließlich verfügt das System noch über eine Datenübertragungseinheit, an die über Telefonleitungen ein Computer oder andere Steuergeräte angeschlossen werden können.

Das kommerzielle Modell 5TI arbeitet solange gut, wie einfache digitale Ein-/Aus-Eingangs- und Ausgangssignale vorliegen. Bei komplexeren Anforderungen, in Anwendungen mit hohen Geschwindigkeitsanforderungen oder auch bei hohen Stückzahlen sollte eine auf den Kunden zugeschnittene Mikroprozessor-Steuerung verwendet werden. Peripherie-Steuergeräte sind beispielsweise eine typische Anwendung, die einen solchen Entwurf erfordern. Im vorigen Beispiel diente der interne Takt der Steuereinheit zur zeitlichen Koordinierung der System-Operationen. Bei einem Peripherie-Steuergerät jedoch wird die Mikroprozessor-Logik durch die Taktsignale des zu steuernden Gerätes synchronisiert. Für eine solche Anwendung benötigt der Mikroprozessor Interrupt-Möglichkeiten, um den Arbeitsablauf des

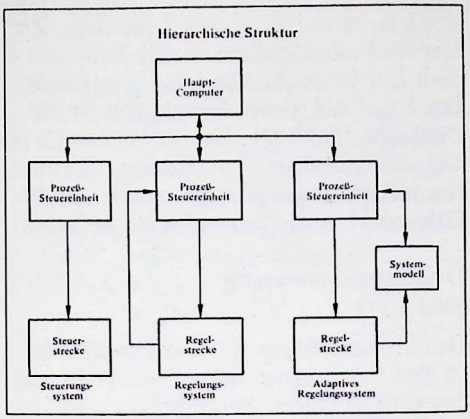


Bild 9.12
Hierarchische Struktur

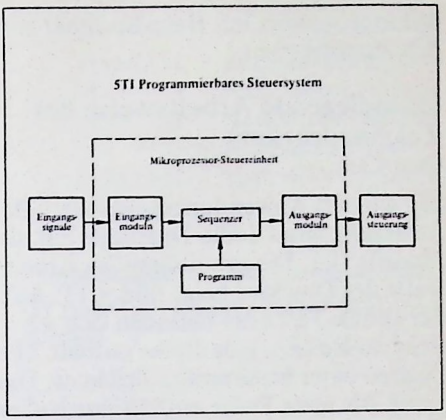


Bild 9.13
5TI Programmierbares Steuerungssystem

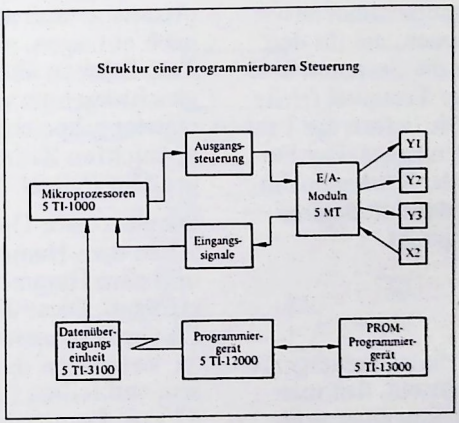


Bild 9.14
Struktur einer programmierbaren Steuerung

Mikroprozessors mit Hauptspeicher zeitlich abzustimmen.

Grundlegende Arbeitsweise des Zeichendruckers
(Bild 9.15)

Ein anderes Anwendungsbeispiel ist die Steuerung eines Seiko-Druckers, z.B. des Modells 101. Die grundsätzliche Arbeitsweise des Druckers zeigt Bild 9.15. Auf der Druck-Trommel befinden sich 42 Zeichen-Reihen, jede Reihe enthält 21 Spalten eines bestimmten Zeichens. Das heißt, die erste Reihe enthält nur Nullen, die zweite nur Einser usw. Wenn sich nun die Trommel dreht, so daß das gewünschte Zeichen unter den Hammer-Kopf gelangt, wird der Anschlagmagnet erregt. In diesem Moment schlägt der Hammer an, und das gewünschte Zeichen ist gedruckt. Die Zeichentrommel enthält die Taktinformationen, die für den Druckvorgang bzw. für die Sensoren und den Ferrit innerhalb der Trommel (siehe Bild) notwendig sind. Sie liefern die Taktsignale für den Hammermagnet. Der Ferritkern erzeugt nach jeder Trommel-Umdrehung einen Impuls, der den Beginn der nächsten Zeile ankündigt.

Druckvorgang
(Bild 9.16)

Die Anforderungen an dieses Steuergerät lassen sich dadurch ermitteln, daß man die Vorgänge beim Drucken einer typischen Zeile betrachtet. Wenn sich in Bild 9.16 die 1 unter den Hämmern befindet, werden beide Einser in ihrer korrekten Position gedruckt, wobei die Hämmer 1 und 7 erregt werden. Das gleiche geschieht bei den zwei Dreiern: die Hämmer schlagen an den Positionen 2 und 6 an usw., bis schließlich die ganze Zeile gedruckt ist. Wenn sich die Zeichentrommel einmal vollständig gedreht hat, wird in der gleichen Weise mit einer neuen Zeile begonnen. Das Steuergerät muß die Textzeile untersuchen und bestimmen, wo jeder Zeichentyp in der Zeile liegt. Zuerst sind alle Zeichen in der Zeile mit dem Zeichencode für 1 zu vergleichen. Das Ergebnis dieses Vergleichs ist die Dualzahl 1000001. Damit ist bereits genau das Steuerwort gefunden, das zum Hammer-Erreger gesandt wird, um die Hämmer 1 und 7 anschlagen zu lassen.

Druckersteuerung
(Bild 9.17)

Das Steuerprinzip für den Drucker ist in Bild 9.17 dargestellt. Die Zeichencodes sind in einem Zeichengenerator ROM oder PROM gespeichert. Das Textspeicher-Register oder -RAM wird bei jeder Zeile auf den neuesten Stand gebracht. Der Mikroprozessor liest aus dem Zeichengenerator -ROM den zu vergleichenden Zeichencode aus. Der Mikroprozessor muß außerdem das RAM danach abfragen, welche Zeichen mit dem Test-Zeichen übereinstimmen. Das Vergleichsergebnis wird in einem Ausgaberegister gespeichert, das die Länge der gedruckten Zeile haben muß, in vorliegendem Fall 21 Bits.

Die Prozessor-Operationen müssen zwischen dem Hammer-Abfallimpuls, TL, und dem Hammer-Erregerimpuls, TP, erfolgen. Diese Zeit beträgt etwa 40 ms. Da vom Prozessor in weniger als 40 ms 21 Vergleiche durchgeführt werden müssen, verbleiben für ein Operationszyklus 525 μs. Damit ergeben sich für die meisten der heute angebotenen Prozessoren keine Probleme.

TL wird als Interrupt-Signal verwendet, um das Unterprogramm "Erzeuge den Ausgangs-Dualcode" in Gang zu setzen. TP läßt diesen Code zu den Hämmer-Magneten passieren, und R, der Rücksetz- oder Trommel-Umdrehungsimpuls, gibt die nächste Zeile für das RAM frei.

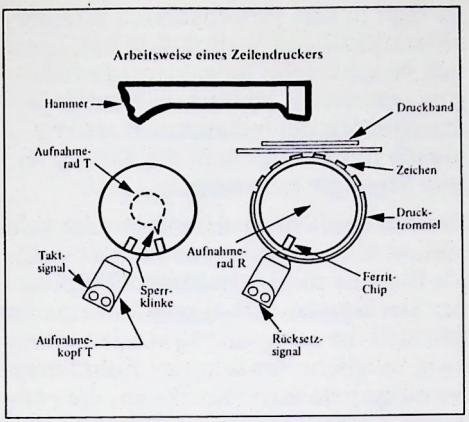


Bild 9.15
Arbeitsweise eines Zeilendruckers

13A7A31	Zu druckende Zeile
1 1	Nach zweiter Position
13 31	Nach vierter Position
13 7 31	Nach achter Position
13A7A31	Nach elfter Postiion
. . . . .	
. . . . .	
.....0001000001	Hammer-Steuerwort

(21 Positionen; füllende Nullen nicht gezeigt)

Bild 9.16
Druckvorgang

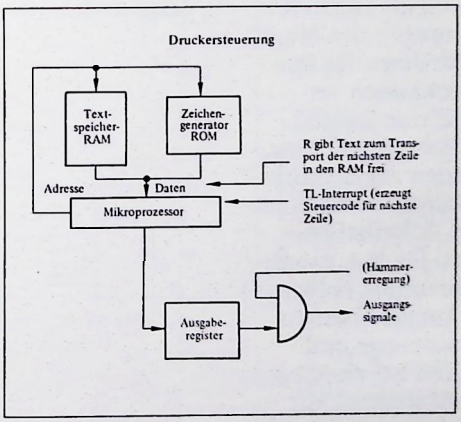


Bild 9.17
Druckersteuerung

Vergleichsbefehl
(Bild 9.18)

Die vom Mikroprozessor auszuführenden Befehle und Operationen sind letztlich auf den Vergleichsbefehl beschränkt (Bild 9.18). Er vergleicht nacheinander den momentanen Zeichencode mit jedem Test-Zeichencode. Bei Gleichheit wird eine "1" erzeugt, die in das Ausgabe-Speicherregister geschoben wird. Nach 21 Vergleichen wird die Textzeile gedruckt.

Renner-Forschungsstation
(Bild 9.19)

Bis jetzt wurde anhand zweier Beispiele gezeigt, wie bestehende Geräte ein naheliegendes Problem lösen können. Bei hochspezialisierten Prozeßsteuerungen befinden sich die Entwickler in keiner so glücklichen Lage, weil die Hersteller von integrierten Schaltungen den Markt bevorzugen müssen, der ihnen für ihre Bauelemente hohe Stückzahlen verspricht. Ein Beispiel für eine äußerst spezielle, einzigartige Prozeßsteuerung, die einen auf den Kunden zugeschnittenen Entwurf einer Mikroprozessor-Steuerung erfordert, ist ein Ackerberieselungs-Steuerungssystem für eine experimentelle Pflanzenzuchtstation entsprechend Bild 9.19. Die Aufgabe besteht darin, erstens die Wassermenge und zweitens die Feuchtigkeit bei einer bestimmten Bodentiefe zu steuern. Mit einem solchen System läßt sich die geeignetste Bewässerungs-Prozedur ermitteln, um die Zuchtqualität bei einem Minimum an Wasser zu optimieren.

Experimentelle Pflanzenzuchtstation
(Bild 9.20)

Wie aus Bild 9.20 ersichtlich, ist der Boden der Station in verschiedene Streifen aufgeteilt, in denen je nach Bewässerungsstrategie ein unterschiedlicher Feuchtigkeitsgrad aufrechterhalten wird.

Da man in den verschiedenen Streifen unterschiedliche Stellpunkte hat, lassen sich verschiedene Bewässerungsstrategien realisieren. Wenn man nun für jeden Streifen den gesamten Wasserverbrauch mißt, kann man die Kosten für jede Strategie ermitteln.

Die von den Forschern mitgeteilte Information über die Pflanzenqualität stellt die Eingabe für die Leistungsfähigkeit der Streifen dar. Mit dieser Information läßt sich die optimale Strategie bestimmen, möglicherweise unter Zuhilfenahme einiger statistischer Daten, die von einem Zentralrechner aufbereitet wurden.

Streifenauslegung
(Bild 9.21)

Die Streifenauslegung ist in Bild 9.21 dargestellt.

Ausgangssignal 0
11001011 > 11001010

Ausgangssignal 1
11001011 = 11001011

Ausgangssignal 0
11001011 < 11000000

Bild 9.18
Vergleichsbefehl

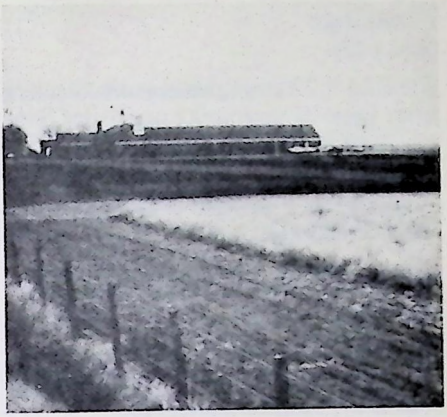


Bild 9.19
Renner-Forschungsstation

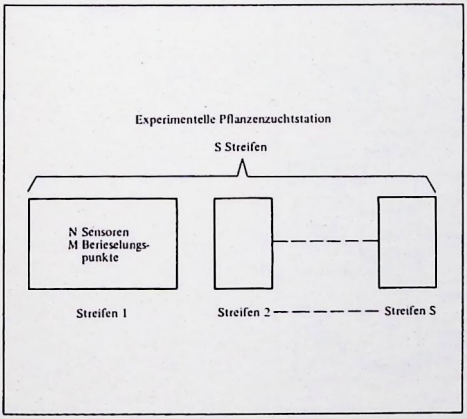


Bild 9.20
Experimentelle Pflanzenzuchtstation

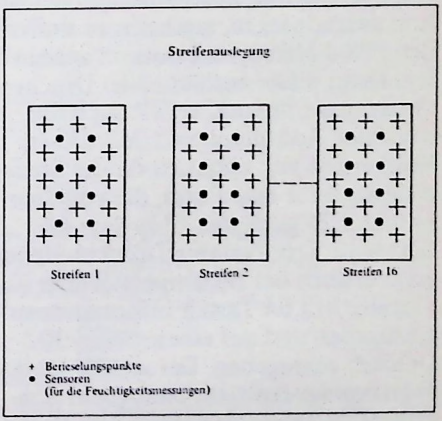


Bild 9.21
Streifenauslegung

Wortlänge
(Bild 9.22)

Wie Bild 9.22 zeigt, kann jeder Streifen, jeder Berieselungspunkt und jeder Sensor durch individuelle 4-Bit-Codes adressiert werden. Außerdem sind die Sensorpegel und die Pflanzenqualität in 4-Bit-Codes umgesetzt.

Mikroprozessor-Spezifikationen
(Bild 9.23)

Die Anforderungen an den Mikroprozessor für dieses Steuerungssystem sind in Bild 9.23 zusammengefaßt.

Steuerung für die experimentelle Pflanzenzuchtstation
(Bild 9.24)

Da es wünschenswert ist, die Ergebnisse von System- und Zustandsdaten der Station auszudrucken, erscheint es sinnvoll, den PPS-4-Mikroprozessorsatz auszuwählen. Denn dieser enthält einen Druckersteuerungs-Chip und erfüllt auch die sonstigen Anforderungen. Mit dieser Wahl ergibt sich natürlich für die Steuerungsstruktur eine Form, die von dem in Bild 9.24 gezeigten Chip-Satz diktiert wird. Das Tastatur/Drucker-Steuergerät erlaubt der Bedienperson über eine Tastatur mit 64 Tasten Informationen einzugeben und auf einem Seiko-302-Drucker auszugeben. Der serielle Datenübertragungs-Baustein ermöglicht den Zugriff zu Telefonleitungen, um Daten an einen Zentralcomputer zu übertragen. Die einzuschaltenden Berieselungspunkte werden durch die Decodier/Latch-Schaltung ausgewählt.

Zusammenfassung

Mit einer Mikroprozessor-Steuerung läßt sich der ausführliche Hardware-Entwurf einer Steuerung beträchtlich vereinfachen. Der Aufwand für den Systementwurf besteht hauptsächlich in der Systemdefinition, der Mikroprozessor-Auswahl und in der Verbindung der einzelnen Chips einer Mikroprozessor-Familie, die für die beabsichtigte Aufgabe nötig sind. Was dann noch verbleibt ist mit Hilfe von Software das System betriebsfähig zu machen. Obwohl es so aussieht, als ob die Software die Hauptschwierigkeit darstelle, werden sich die meisten Entwickler nach einem vollständigen Entwurfsprozeß gewiß wundern, warum Mikroprozessoren und ihre vielseitige Software nicht schon früher verfügbar waren.

16 Streifen, Code	=	4 Bits
15 Sensoren, Code	<	4 Bits
8 Berieselungs-punkte, Code	<	4 Bits

Bild 9.22
Wortlänge-Anforderungen

- 4-Bit-Struktur
- Ausführungszeiten im ms-Bereich
- 16 x 23 = 368 Eingangs-/Ausgangs-Ports (ca. 1 Bit)
- Interruptmöglichkeiten
- Etwa 8K Speicher-ROM und RAM
- Einfaches Drucker-Interface

Bild 9.23
Mikroprozessor-Spezifikationen

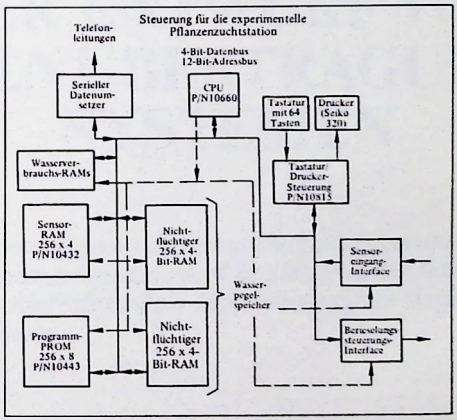


Bild 9.24
Steuerung für die experimentelle Pflanzenzuchtstation

KAPITEL 10

MIKROPROZESSOR-ANWENDUNGEN: DER ENTWURF VON DATENÜBERTRAGUNGS-SYSTEMEN

Neben der Besprechung von digitalen Datenübertragungsproblemen und typischen Hardware-Lösungen wird in diesem Kapitel gezeigt, wie in diesem breiten Anwendungsfeld Mikroprozessoren wirtschaftlich eingesetzt werden können.

Mikroprozessoranwendungen: Der Entwurf von Datenübertragungssystemen

Dieses Kapitel untersucht ein anderes potentielles Anwendungsgebiet für Mikroprozessoren, die Verwendung des Mikroprozessors als Datenübertragungs-Prozessor. Der Zweck eines Datenübertragungs-Prozessor-Systems wird analysiert und viele der Probleme, die uns bei der Implementierung solch eines Systems begegnen, umrissen. Schließlich wird der Entwurf eines Datenübertragungs-Prozessors beschrieben.

Computersystemanwendungen erfordern Datenfernübertragung (Bild 10.1)

Die Datenübertragung ist heute eines der am schnellsten wachsenden Gebiete in der Computerindustrie. Dieses Wachstum hat seine Ursache darin, daß viele Computersysteme entwickelt wurden, die die Computerleistung auch weit entfernten Anwendern zur Verfügung stellen. In Bild 10.1 sind einige dieser Anwendungen aufgelistet. Alle diese Systeme erfordern eine hochentwickelte Datenfernübertragung zwischen den weit entfernt aufgestellten Terminals (Datenanschlußstation bzw. Datenendstation) und dem Hauptrechner. Die Datenübertragung über Datenfernübertragungssysteme zwischen Terminals und Zentralcomputer ist weit komplexer, als nur die Erstellung einer Verbindung zwischen beiden Einheiten.

Typisches Datenfernübertragungssystem zwischen entfernt aufgestellten Terminals und Zentralcomputer (Bild 10.2)

Wie Bild 10.2 andeutet, ist der erste Grund für die Problematik der Datenfernübertragung zwischen entfernten Terminals und Zentralcomputer die Verwendung gewöhnlicher Telefonleitungen für die Datenübertragung. Telefonleitungen und -schaltungen sind analog, während Computer digital arbeiten.

Werden analoge Telefonnetze für die Übertragung digitaler Signale verwendet, müssen die digitalen Signale jeweils am Anfang der Leitung in analoge und am Ende wieder in digitale umgesetzt werden. Das Gerät, das diese Digital/Analog-Umsetzung durchführt, wird MODEM genannt. MODEM ist eine Abkürzung für MOdulator und DEModulator.

- Flug-Reservierung
- Bankkontenführung
- Verkaufsstellen-Terminals
- Börsen-Information
- Hotel- und Mietwagen-Reservierung
- Dezentrale Datenverarbeitung
  - Timesharing
  - Remote Batch
- Dezentrale Datenerfassung

Bild 10.1
Computersystemanwendungen, die Datenfernübertragung erfordern

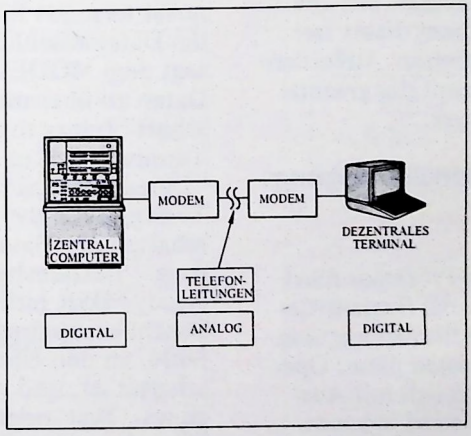


Bild 10.2
Ein typisches Fern-Kommunikationssystem zwischen dezentralen Terminals und einem Zentralcomputer

Grundsystem mit einem Datenübertragungs-Prozessor zwischen Zentralcomputer und MODEM
(Bild 10.3)

Die längste Zeitspanne, die das MODEM für seine Aufgaben verbraucht, betrifft nicht den eigentlichen Datentransfer, sondern Aufgaben wie: Abfrage, Kontrolle, Leitungsumschaltung, Fehlererkennung und wiederholte (Mehrfach-) Übertragung. Die meisten dieser Aufgaben verlangen nicht die Leistungsfähigkeit der CPU eines Computers. Überträgt man aber trotzdem die Behandlung dieser Aufgaben der CPU, verliert diese eine Menge der Zeit, die sonst für die Verarbeitung der von den verschiedenen Terminals ankommenden Informationen zur Verfügung stehen würde. Daraus folgt, wie Bild 10.3 zeigt, daß die beste Lösung die Vorschaltung eines Datenübertragungs-Prozessors vor den Zentralcomputer ist, damit dieser frei wird für sinnvollere Arbeiten. Außerdem wird durch so ein Konzept das gesamte System wesentlich billiger.

Aufgaben eines Datenübertragungs-Prozessors
(Bild 10.4)

Der Datenübertragungs-Prozessor führt viele, sehr wichtige, für das System "lebensnotwendige" Operationen aus, wie Bild 10.4 zeigt. Die meisten dieser Operationen sind wahrscheinlich mit Ausnahme der Prozedur bestens bekannt. Als Prozedur ist leider nur der definierte Ablauf für die Operationen zwischen dem Computer und dem MODEM zu verstehen. Glücklicherweise ist die Schnittstelle zwischen Rechner und Datenübertragungs-Netzwerk im EIA-Standard-RS232C festgelegt. Diese Standardschnittstelle läßt sich anwenden für: asynchrone Übertragung und synchrone Übertragung, im Wählnetz oder mit Standleitung im Simplex-, Halb-Duplex- und Voll-Duplex-Betrieb.

Die Spezifikationen für die verschiedenen Signale können in vier Gruppen eingeteilt werden: Masseleitungen, Steuersignale, Zeit- und Datensignale. Zwanzig Signaltypen sind in den RS232-Spezifikationen festgelegt; sie sind am Schluß dieses Kapitels definiert.

Masseleitungen
(Bild 10.5)

Wie aus Bild 10.5 hervorgeht, ist die Schutzerde (-masse) die Erdung des Gehäuses (Chassis) im MODEM. Die Betriebserde legt das allgemeine Masse-Referenz-Potential für alle Schnittstellenschaltkreise fest.

Steuersignale
(Bild 10.6)

Die Steuersignale sind in Bild 10.6 aufgeführt. "Sendeteil einschalten" (Request to send) RTS wird von der lokalen Datenanschlußstelle erzeugt und sagt dem MODEM, daß der Computer Daten zu übermitteln hat. "Sendebereitschaft" (clear to send) CTS sagt dem Terminal, daß es Daten senden kann. "Datenterminal betriebsbereit" (Data Terminal Ready) DTR steuert das Umschalten der Datenübertragungseinrichtung. "Betriebsbereitschaft" (Data Set Ready) DSR indiziert, daß das lokale Datenübertragungsnetzwerk, das MODEM, an den Übertragungskanal eingeschaltet ist, und sich nicht im Übertragungs-, Test- oder Wählmodus befindet. "Empfangssignal" (Data carrier detect) DCD wird vom MODEM erzeugt und bedeutet, daß das empfangene Signal gültige Daten enthält. Der "ankommende Ruf", vom MODEM erzeugt, zeigt an, daß das MODEM angewählt wurde.

Daten, die letzte der vier Signal-Kategorien, werden von den Datenanschluß-(end-)Stationen in geeigneter Weise erzeugt und dem MODEM übertragen.

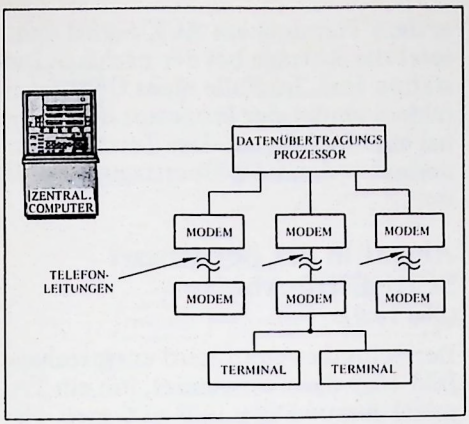


Bild 10.3
Grundsystem mit einem Datenübertragungs-Prozessor zwischen Zentralcomputer und MODEM

- Datenübertragungsprozedur
- Datenübertragung
- Datenwiederholung
- Fehlererkennung und -Korrektur
- Code-Umsetzung
- System-Aktivitäts- und Fehlerstatistik

Bild 10.4
Aufgaben eines Datenübertragungs-Prozessors

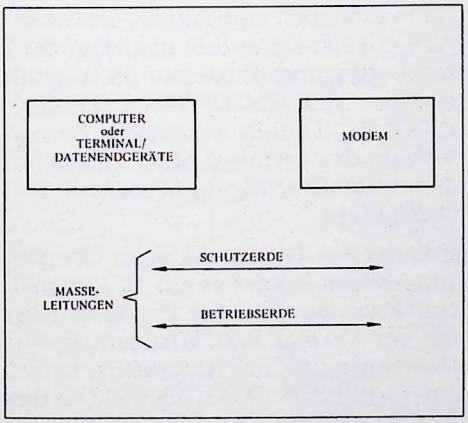


Bild 10.5
Masseleitungen

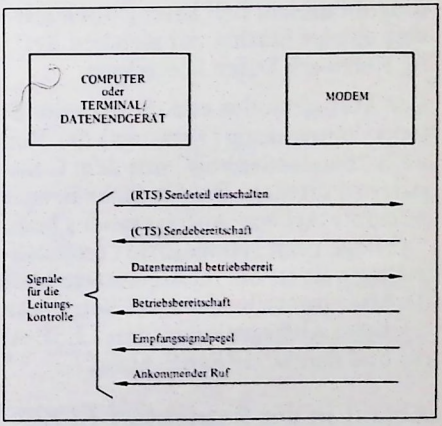


Bild 10.6
Steuersignale

Der Prozessor muß alle diese Signale erkennen und überwachen bzw. erzeugen, um sicherzustellen, daß sie innerhalb der vorgeschriebenen Zeitspanne und in der richtigen Reihenfolge eintreffen. Zusätzlich zur normalen Leitungsprozedur muß der Datenübertragungsprozessor in der Lage sein, auf Anrufe selbstständig zu antworten und selbstständig die Leitungsverbindungen anzuwählen.

Grundstruktur eines Terminal-Netzwerkes
(Bild 10.7)

Eine andere wichtige Funktion des Datenübertragungs-Prozessors ist das Abfragen von Stationen im Netzwerk. Dies ist in Bild 10.7 skizziert. Das Abfragen von Stationen ist notwendig, um das Netzwerk kostenmäßig gut auszunützen. Hierbei schaltet man so viele Terminals wie möglich und notwendig an eine Leitung. In diesem Fall kann jedoch nur eine einzige Station zur gleichen Zeit im Netzwerk Daten übertragen.

Der Abfragemodus erlaubt es, jeder Datenendeinrichtung (Terminal) der Reihe nach "Sendeerlaubnis" mit dem Computer zu erteilen. Die für unser Beispiel genannte Art von Abfragemodus heißt "Abfrage nach Namensliste" (roll call polling); sie ist die meistbenutzte Methode. Man unterscheidet zwei Betriebsarten beim Abfragemodus, den LESE-Abruf und den SCHREIBE-Abruf.

Ablauf in der Betriebsart LESE-Abruf
(Bild 10.8)

Wie in Bild 10.8 zu sehen ist, wird beim Lese-Abruf durch Senden von EOT-ID-END-Signal beim jeweiligen Terminal nachgefragt, ob dieses irgendeine Nachricht zum Senden bereithält. Wenn nicht, antwortet das Terminal mit einem NAK-Signal; liegt eine Meldung vor, wird diese einfach gesendet. Empfängt der Prozessor die Nachricht ohne Fehler, sendet er dem Terminal ein ACK-Signal und setzt die Abfrage bei der nächsten Datenstation fort. Im Falle eines Übertragungsfehlers sendet der Prozessor dem Terminal ein NAK-Signal. Das Terminal wiederholt sodann die Übertragung der Daten.

Ablauf in der Betriebsart SCHREIBE-Abruf
(Bild 10.9)

Der Schreibe-Abruf wird entsprechend Bild 10.9 dazu verwendet, um ein Terminal auszuwählen und zu fragen, ob dieses in der Lage ist, die Meldung, die der Datenübertragungs-Prozessor bereit hält, zu empfangen. Der Datenübertragungs-Prozessor frägt das Terminal dadurch ab, daß er ein EOT-ID-STX-Signal sendet. Ist das Terminal beschäftigt und kann es die Meldung daher nicht akzeptieren, sendet es ein NAK-Signal zurück. Kann es jedoch empfangen, sendet es ein ACK-Signal. Daraufhin überträgt der Datenübertragungs-Prozessor im folgenden Format: STX-ID-CONTROL-TEXT-EXT-LRC. Dies ist das gleiche Format, welches das Terminal benutzt, um Daten dem Datenübertragungs-Prozessor zu übermitteln.

Erkennt das Terminal keinen Übertragungsfehler, sendet es ein ACK-Signal zum Prozessor und der Prozessor fährt mit der Abfrage fort. Wird jedoch ein Übertragungsfehler festgestellt, sendet das Terminal ein NAK-Signal. Der Prozessor wiederholt dann gemäß einer vorher festgelegten Anzahl die Übertragung und meldet schließlich dem Bedienpersonal, daß die Verbindung zusammengebrochen ist.

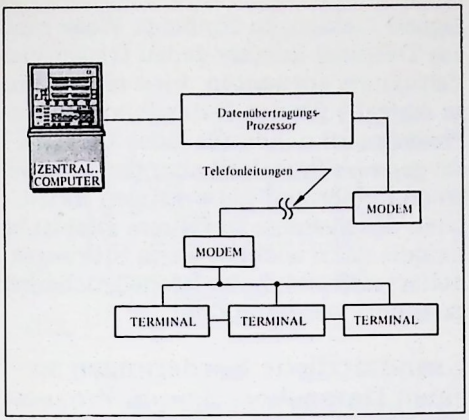


Bild 10.7
Grundstruktur eines Terminal-Netzwerkes

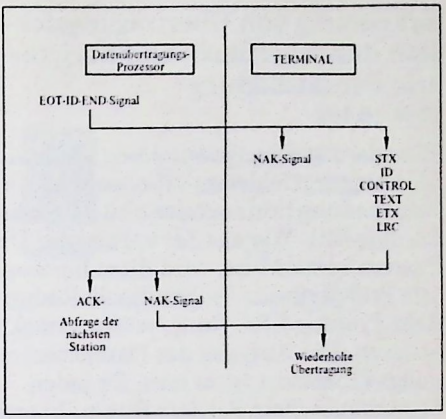


Bild 10.8
Ablauf in der Betriebsart Lese-Abruf

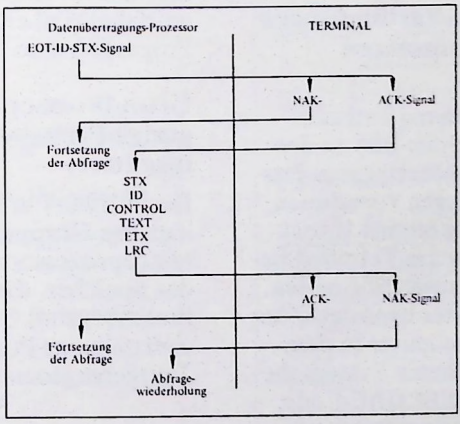


Bild 10.9
Ablauf in der Betriebsart Schreibe-Abruf

Erkennung von Übertragungsfehlern durch vertikale und horizontale Paritätsbildung
(Bild 10.10)

Eine der häufig angewendeten Methoden, Übertragungsfehler zu erkennen, ist die Verwendung horizontaler und vertikaler Paritäts-Bits. Wie aus der vorherigen Diskussion bekannt ist, wird diese horizontale Prüf-Methode longitudinale Redundanz-Prüfung LRC (long, redun. check) genannt. Die Aufgabe des Datenübertragungs-Prozessors ist es nun, für jeden Datenblock diese Paritäts-Bits zu berechnen. Das Terminal errechnet die Paritäts-Bits für die empfangenen Daten noch einmal und leitet im Falle einer Abweichung die wiederholte Übertragung des Datenblocks ein.

Datenübertragungs-Verbindungen müssen den Code umsetzen
(Bild 10.11)

Zusätzlich zur Berechnung und zum Hinzufügen dieser Paritäts-Bits zu den Daten muß der Datenübertragungs-Prozessor Code-Umsetzungen vornehmen, und zwar wechselseitig einmal Umsetzung in den Code, den das Terminal benützt, zum anderen Umsetzung in den Code, den der Computer benötigt. Zum Beispiel sendet der Computer in dem im Bild 10.11 gezeichneten System die Information im 8-Bit-EBCIDIC-Code, den der Datenübertragungs-Prozessor in den 5-Bit-Baudot-Code mit "gerader" Parität umrechnen muß, da das Terminal nur diesen Code weiterverarbeiten kann.

Ein anderer Fehlertyp, den der Prozessor prüfen muß, ist die sogenannte Zeit-Überschreitung (time out). Einige wenige Beispiele sind: Das MODEM muß das CTS-Signal innerhalb typisch 150 Millisekunden nach dem RTS-Signal erzeugen. Passiert dies nicht innerhalb dieser Zeitspanne, muß der Prozessor dem Bedienpersonal das Ausbleiben des Signals melden. In ähnlicher Weise muß das Terminal in einer genau festgelegten Zeitspanne antworten. Eine sehr wichtige Aufgabe für den Datenübertragungs-Prozessor ist daher, für jedes Terminal ein genaues Protokoll über die aufgetretenen Fehler und die sonstigen Aktivitäten des Systems zu führen. Dies ist besonders dann wichtig, wenn Störungen isoliert und eine hohe Betriebssicherheit garantiert werden sollen.

Grundsätzliche Forderungen an einen Datenübertragungs-Prozessor
(Bild 10.12)

Beim Entwurf eines Datenübertragungs-Prozessors mit Hilfe eines Mikroprozessors ist der erste Schritt die Festlegung der Aufgaben, die das Gerät bewältigen soll. Die Grundforderungen für ein Entwurfsbeispiel eines Datenübertragungs-Prozessors sind in Bild 10.12 aufgeführt.

Grundkonzept eines Datenübertragungs-Prozessors
(Bild 10.13)

Der Entwurf in Bild 10.13 kann in vier logische Gruppen eingeteilt werden: der Mikroprozessor und sein Taktgenerator, der Speicher, die Schnittstelle zwischen Prozessor und Computer und die Datenübertragungs-Ports und der dazugehörige Taktgenerator.

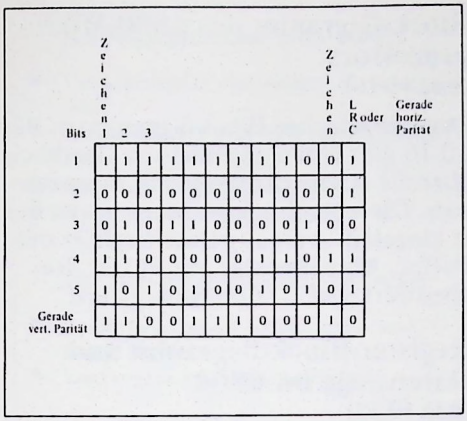


Bild 10.10
Erkennung von Übertragungsfehlern durch horizontale und vertikale Paritätsbildung

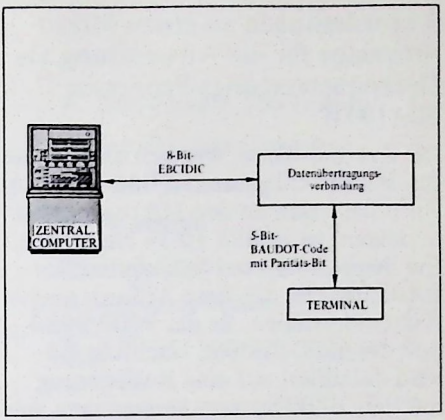


Bild 10.11
Erforderliche Code-Umsetzung der Datenübertragungsverbindung

- Datenvorverarbeitung
- Datentransfer in den Zentralcomputer über dessen DMA-Zugriff
- Pufferspeicher für jeden Datenblock
- Abfrage der Datenendeinrichtungen
- Code-Umsetzung
- Per Programm wählbare Datenrate (Baud-Rate = Bit/s)
- Automatische Beantwortung der ankommenden Anrufe

Bild 10.12
Grundforderungen an einen Datenübertragungs-Prozessor

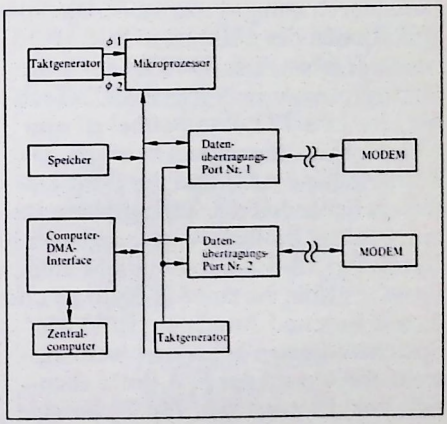


Bild 10.13
Grundkonzept eines Datenübertragungs-Prozessors

Anforderungen an einen Mikroprozessor für die Anwendung als Datenübertragungs-Prozessor
(Bild 10.14)

Die Art und Weise, wie sich die Parameter des Blockdiagramms (Bild 10.13) in Anforderungen an den Mikroprozessor umsetzen, ist in Bild 10.14 angegeben. Ein Beispiel für zwei Mikroprozessor-Bausteinsätze, die diese Anforderungen recht gut erfüllen, ist das 9900/9980- und das 6800-System. Nachfolgend wird detailliert auf eine Realisierung mit dem 6800-System eingegangen werden.

Spezifikationen des 6800-Mikroprozessors
(Bild 10.15)

Der Mikroprozessor ist der Kern des Datenübertragungs-Prozessors. Die Spezifikationen des 6800 ist in Bild 10.15 wiedergegeben. Der 6800 ist ein 8-Bit-Mikroprozessor in N-Kanal-MOS-Technik, der LPS-TTL-kompatibel ist und eine +5 Volt-Versorgung benötigt. Die 8-Bit-Parallele ALU und der 8-Bit-Datenbus behandelt alle auftretenden Datencodes als Einfach-Worte, sogenannte Bytes. Der 16-Bit-Adreßbus läßt eine Speichergröße bis zu 65 K-Byte zu und da alle Ein- und Ausgänge (E/A) wie Speicheradressen behandelt werden, stellt die Anzahl der E/A-Ports ebenfalls kein Problem dar. Die 72 Befehle variabler Länge, der Stack-Speicher variabler Länge und vier Unterbrechungsebenen ermöglichen ein leichtes Programmieren des Betriebssystems.

Der 6800 verfügt außerdem über sieben Adressierungsarten. Der Stack-Speicher wird über einen "Stackpointer", einem 16-Bit-Register auf dem Mikroprozessor (stackpointer) sowie einem beliebig großen RAM-Speicher extern realisiert. Ferner sind sechs interne Register und Interruptvektoren vorhanden.

Blockdiagramm des 6800-Mikroprozessors
(Bild 10.16)

Das vereinfachte Blockdiagramm in Bild 10.16 gibt einen allgemeinen Überblick über die Architektur des Mikroprozessors. Diese Architektur kann in vier Teile eingeteilt werden: Datenregister mit Puffer, Adreßregister mit Puffer, Rechenwerk (ALU) und Steuerwerk.

Register-Blockdiagramm und Datenwege im 6800
(Bild 10.17)

Wie man in Bild 10.17 sieht, enthält der 6800 sechs interne Register, zwei 8-Bit Akkumulatoren, ein 16-Bit Indexregister, einen 16-Bit Programmschrittzähler, ein 16-Bit Stackpointer-Register für den Stack-Speicher und ein 8-Bit Statusregister. Alle sind per Programm ansprechbar.

Der Mikroprozessor sollte haben:

- Vollständigen, vielseitigen Bausteinsatz
- Große Anzahl von EIN-/AUSGABE-Ports
- Interruptmöglichkeiten
- Angemessene Speichergröße

Der Mikroprozessor benötigt keine:

- Leistungsstarken Arithmetik- und Logik-Befehle
- Umfangreiche Softwarehilfen

Bild 10.14
Anforderungen an einen Mikroprozessor für die Verwendung als Datenübertragungs-Prozessor

- 8-Bit-Parallele Verarbeitung
- Bidirektionaler Datenbus
- 16-Bit-Adress-Bus . . . 65k Bytes
- 72 Befehle
- Stack-Speicher variable Länge
- 4 Interruptebenen
- 6 Interne Register

Bild 10.15
6800 Mikroprozessor-Spezifikationen

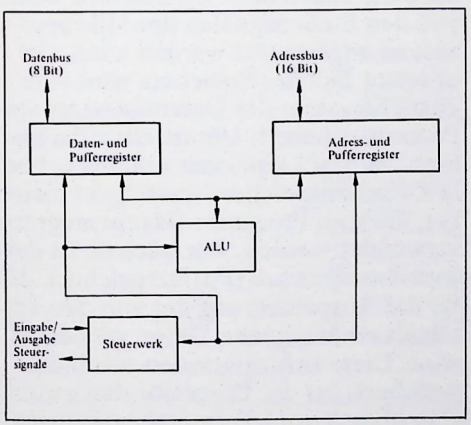


Bild 10.16
Blockdiagramm des 6800-Mikroprozessors

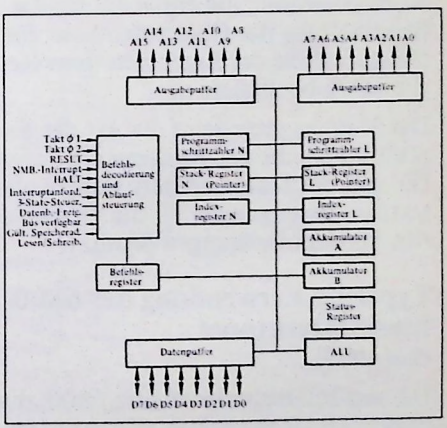


Bild 10.17
Blockdiagramm der Register und Datenwege im 6800

Registerstruktur des 6800
(Bild 10.18)

Der Programmschrittzähler ist, wie Bild 10.18 zeigt, ein 2 Byte = 16 Bit-Register, das auf die augenblickliche Programmadresse zeigt. Sein Inhalt wird nach der Ausführung eines Befehles um 1 inkrementiert oder dekrementiert.

Das Stackpointer-Register für den Stack-Speicher ist ein 2 Byte Register, das die Adresse der nächsten freien Zelle im externen Stack-Speicher enthält. Dieser Stack-Speicher ist im Normalfall ein geschickt gewählter Bereich im Schreib/Lesespeicher (RAM).

Das Indexregister ist ein 2 Byte Register, das für die Speicherung der 16-Bit Adresse im Falle der indizierten Adressierung des Speichers benutzt wird.

Der Mikroprozessor enthält zwei 8-Bit Akkumulatoren, die entweder für die Bereithaltung der Operanden bzw. für die Aufnahme der Ergebnisse vom Rechenwerk bereitstehen.

Das Statusregister zeigt die Art der Ergebnisse an, die das Rechenwerk liefert. Die Bits des Statusregisters werden als testbare Bedingungen für die verschiedenen Sprunganweisungen benutzt.

Typische Verwendung der 6800-Interruptvektoren
(Bild 10.19)

Die vier Interruptebenen des 6800, die in Bild 10.19 angegeben sind, werden für den Restart und die Interruptsteuerung benutzt. Die adressen sind in hexadezimaler Form dargestellt.

Grundsätzliche Speicherorganisation für einen Datenübertragungs-Prozessor
(Bild 10.20)

Der Speicher des Datenübertragungs-Prozessors muß einer Vielzahl von Forderungen angepaßt sein. Er kann in sechs funktionell verschiedene Bereiche eingeteilt werden (Bild 10.20). Der Festwertspeicher (ROM) befindet sich am Ende des Speichers. Er enthält die Interruptvektoren (wie vorher beschrieben) und das Ladeprogramm (Loader). Da das Ladeprogramm und die Interruptvektoren im Festwertspeicher (ROM) abgelegt sind, werden sie im Falle eines Netzausfalles nicht zerstört. Ein Interrupt mit Vektor, der sog. Restart, startet das Ladeprogramm. Das Ladeprogramm lädt das Hauptprogramm für den Datenübertragungs-Prozessor vom Zentralcomputer in den unteren Speicherbereich (RAM), wenn der Restart ausgeführt wird.

Der untere Teil des Speichers besteht aus dem RAM-Speicher. Es ist sinnvoll, ein statisches Standard-MOS-RAM zu wählen (z.B. TMS4043/4042), da dieses ohne zusätzliche Schaltkreise direkt von den Steuersignalen des Mikroprozessors angesteuert werden kann. Der unterste Teil des Speichers wird vom Betriebssystem des Datenübertragungs-Prozessors belegt. Unmittelbar darüber liegt der Stack-Speicher und der schnelle Zwischenspeicher (scratchpad memory), die vom Prozessor (Hauptprogramm) verwendet werden. Der nächste ist der Zwischenspeicher (Pufferspeicher), der für die Abspeicherung der von den Terminals empfangenen Daten verwendet wird. Diese Informationen werden gespeichert, bis der Prozessor den ganzen Datenblock vom Terminal empfangen und auf Übertragungsfehler überprüft hat. Danach werden die Daten dem Computer übermittelt.

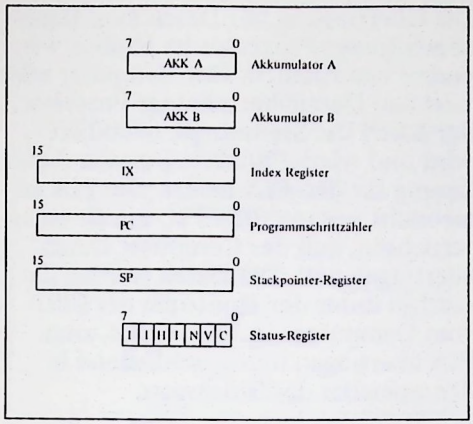


Bild 10.18
Registerstruktur des 6800

VEKTOR MS	VEKTOR LS	Beschreibung
FFFE	FFFF	Wiederstart (RESTART)
FFFC	FFFD	Nichtmaskierbarer Interrupt
FFFA	FFFB	Software Interrupt
FFF8	FFF9	Interruptanforderung

Bild 10.19
Typische Verwendung der 6800-Interrupt-vektoren

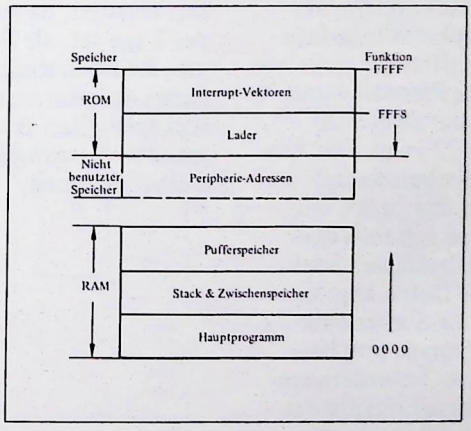


Bild 10.20
Basis-Speicher des Prozessors

Mikroprozessor-Speicherschnittstelle
(Bild 10.21)

Die Wahl eines vielseitigen Bausteinsatzes befreit den Benutzer vom Entwurf diverser Schnittstellen (Interfaces) zwischen den einzelnen Teilen des Mikroprozessors.

So ist zum Beispiel die Speicherschnittstelle, wie Bild 10.21 zeigt, bestehend aus 16 Adressleitungen, 8 bidirektionalen Datenleitungen und 2 Steuerleitungen, bereits voll definiert und erfordert kaum noch Entwurfsarbeit.

Schnittstelle zwischen dem Datenübertragungs-Prozessor und dem Kanal des Zentralcomputers
(Bild 10.22)

Um den Prozessor mit dem DMA-Kanal des Zentralcomputers zu koppeln, verwendet man einen Peripherie-Interfaceadapter (PIA). Die Schnittstelle zwischen dem Datenübertragungs-Prozessor und dem DMA-Kanal benötigt zwei PIA-Bausteine, wie Bild 10.22 zeigt. Der Prozessor lädt den einen der beiden mit dem 16-Bit-Datenwort, das zum Computer übertragen werden soll und den anderen PIA mit der Adresse der Speicherzelle, in welche die Daten abgelegt werden sollen. Erhält die Steuerlogik die Ausgabe-BEREIT-Signale von beiden PIAs, wird ein DMA-Anforderungssignal gesetzt. Dieses Signal signalisiert dem Computer, daß der Prozessor bereit ist, Daten zu übersenden. Ist auch der Computer bereit Daten zu empfangen, wird durch das Signal "DMA-Bestätigung" der Datentransport erlaubt und die Steuerlogik veranlaßt die Abspeicherung der Daten im Speicher. Die Steuerlogik liefert sodann ein Ausgabe-ACK-Signal für den PIA. Daraufhin unterbricht der PIA die CPU und meldet damit der CPU, daß die Übertragung beendet ist.

Die Übertragung der Daten zum Datenübertragungs-Prozessor ist ähnlich wie vorher beschrieben. Der Computer selektiert den Datenübertragungs-Prozessor, der durch die Steuerlogik decodiert wird und wiederum eine Eingabe-Anforderung für den PIA liefert. Der PIA unterbricht sodann die CPU, um sie zu unterrichten, daß der Computer Daten übertragen will. Die Daten werden daraufhin unter der Kontrolle der CPU vom Computer, 16 bit parallel, zum PIA übertragen und anschließend in den Speicher des Prozessors.

Ausgabepufferung für den peripheren Schnittstellen-Adapter (PIA)
(Bild 10.23)

Der Ausgang des PIA muß, wie Bild 10.23 zeigt, vom Datenbus her gepuffert werden, da der Baustein nicht in der Lage ist, als Bus-Treiber zu fungieren. Es ist deshalb wichtig, sich stets daran zu erinnern, daß die Ausgänge von speziellen Bausteinen nur eine begrenzte Ausgangsbelastung (Fan Out) treiben können.

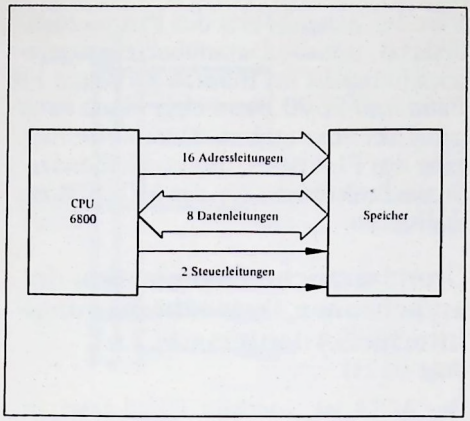


Bild 10.21
Mikroprozessor-Speicherschnittstelle

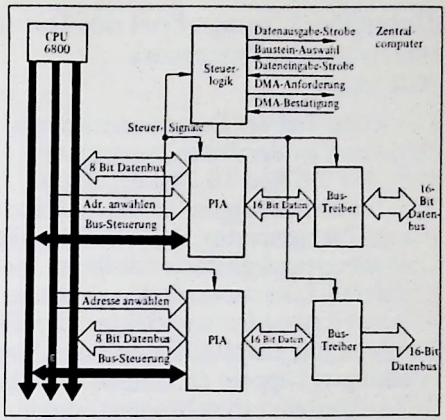


Bild 10.22
Schnittstelle zwischen Datenübertragungs-Prozessor und DMA-Kanal des Zentralrechners

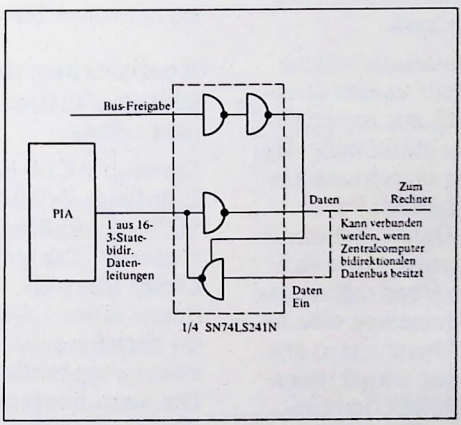


Bild 10.23
Ausgabepufferung für den peripheren Schnittstellen-Adapter (PIA)

Datenübertragungs-Port des Datenübertragungs-Prozessors
(Bild 10.24)

Der letzte Teil des Datenübertragungs-Prozessors ist der Datenübertragungs-Port, wie ihn Bild 10.24 zeigt. Dieser Port besteht aus einem Sende- und Empfangs-Taktgenerator, der alle Teile des Datenübertragungs-Ports mit ihrem spezifizierten Takt versorgt, den programmierbaren Baud-Raten-Selektor; den PIA-Baustein, um das MODEM mit den Steuerleitungen koppeln zu können, und den ACIA-Baustein (asynchronous communication interface adapter), der für die eigentliche Übertragung und den Empfang der Daten verantwortlich ist.

Der asynchrone Datenübertragungs-Interface-Adapter verrichtet viele der Aufgaben, die für den Empfang und die Übertragung von Information von und zum MODEM zu bewältigen sind.

Der ACIA erfüllt vier unterschiedliche Funktionen: erstens stellt er eine direkte Schnittstelle zur CPU, zur Interruptsteuerung und zur Bus-Schnittstelle dar; zweitens versieht er die asynchrone Datenübertragung, einschließlich Parallel-Seriell-Umsetzung der Daten, das Einfügen von Start- und Stopbits sowie Paritätsbits und die serielle Übertragung der Datenworte, die im allgemeinen eine beliebige Anzahl von Bits beinhalten; drittens bewerkstelligt er den asynchronen Datenempfang. Dies schließt den Empfang von seriellen, asynchronen Daten der Standard-Wortlängen, Erkennung von Start- und Stopbits, Paritäts- und Fehlerüberprüfung und seriell-parallele Umsetzung der Daten ein; viertens übernimmt er eine begrenzte Anzahl von Steuerfunktionen, die der PIA nicht abdeckt.

Der periphere Interface-Adapter (PIA) erfüllt zwei Aufgaben. Erstens stellt er einen per Programm wählbaren Sende/Empfangs-Takt für den ACIA bereit. Vier der Ausgabebits des PIA werden benutzt, um die Datenübertragungsgeschwindigkeit im Bereich zwischen 110 Baud und 9600 Baud über einen externen Takt vorzugeben. Die zweite Aufgabe des PIA ist es, die verbleibenden Steuerfunktionen für das MODEM zu realisieren.

Grundsätzliche Organisation des asynchronen Datenübertragungs-Interface-Adapters (ACIA)
(Bild 10.25)

Der ACIA ist, wie Bild 10.25 zeigt, in fünf Teile aufgegliedert. Es sind dies die Sende- und Empfangs-Datenregister, welche die Parallel/Seriell- bzw. Seriell/Parallel-Umsetzung übernehmen, die Takt- und Ablaufsteuerung, das Statusregister und das Steuerregister. Der ACIA ist damit ein nahezu komplettes asynchrones Dateninterface.

Realisierung des Sende- und Empfangs-Zeitbasis
(Bild 10.26)

Die vom ACIA benötigte Sende- und Empfangs-Zeitbasis wird, wie in Bild 10.26 symbolisiert, durch den quarzgesteuerten Taktgeber MC14411 erzeugt. Dieser Baustein teilt die Ausgangsfrequenz eines 1,843226-MHz-Quarzes in 64 verschiedene, für Kommunikationszwecke sinnvolle Frequenzen herunter. Die neun heutzutage gebräuchlichsten Übertragungsgeschwindigkeiten für Terminals sind: 9600, 7200, 4800, 2400, 1200, 300, 150, 134.49 und 110 Bit/s. Die Ausgänge dieses Gerätes werden einem programmierbaren Baud-Raten-Selektor – auf jedem Port vorhanden – zugeführt.

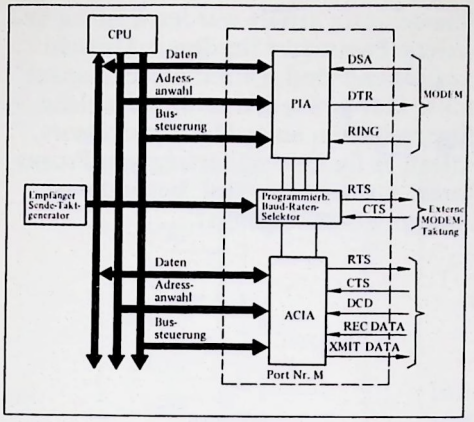


Bild 10.24
Datenübertragungs-Port des Datenübertragungs-Prozessors

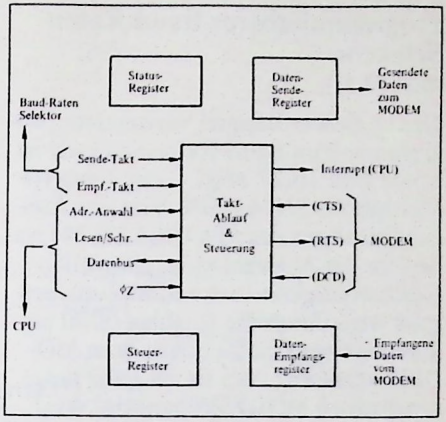


Bild 10.25
Grundsätzliche Organisation des asynchronen Datenübertragungs-Schnittstellenadapters

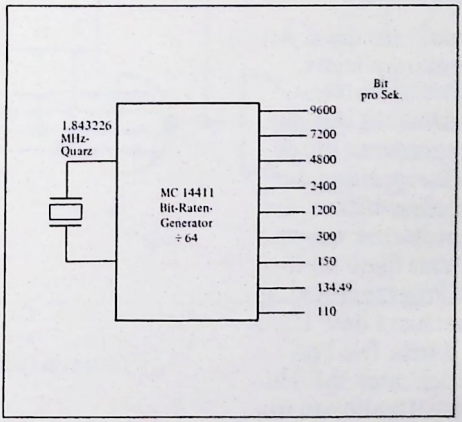


Bild 10.26
Realisierung der Sende- und Empfangs-Zeitbasis

Programmierbarer Baud-Raten-Selektor
(Bild 10.27)

Der in diesem Beispiel verwendete programmierbare Baud-Raten-Selektor ist – wie Bild 10.27 zeigt – ein 1-aus-10-Multiplexer SN74150N. Vier der Ausgabeleitungen des PIA (Bild 10.24) werden für die Auswahl einer sinnvollen Geschwindigkeit verwendet. Zum Beispiel wird durch die Bitfolge 0000 am Steuereingang die Zeitbasis vom MODEM adressiert, wie sie im Falle eines synchronen MODEMS benötigt wird.

Der Datenübertragungs-Port (Bild 10.24) muß für jede Datenübertragungsleitung, die vom Datenübertragungs-Prozessor bedient wird, verdoppelt werden. Die Code-Umsetzung und Fehlererkennung werden durch die Software im Datenübertragungs-Prozessor ausgeführt.

Die Software (Programme) für diese Anwendung ist ziemlich unkompliziert. Die Programmierung wird sinnvollerweise in Assemblersprache und mit Hilfe eines Simulationsprogrammes durchgeführt. Damit ist die Überprüfung der komplexen Schnittstellenprobleme, die früher reine Hardwareprobleme waren, wesentlich einfacher. Wird diese Methode gewählt, sollten die Programmierung und Simulation entsprechend dem Hardware-Aufbau vor sich gehen. Die Programmierung kann jedoch auch mit Hilfe von geringfügigen Modifikationen unabhängig von der Hardware-Verfügbarkeit durchgeführt werden.

Zusammenfassung

In diesem Kapitel konzentrierte sich die Diskussion von Datenübertragungs-Prozessoren auf die Aufgaben derselben und auf die Frage, warum sie benötigt werden. Es wurde an Hand eines Beispiels untersucht, wie die Forderungen, die durch ein solches System entstehen, durch einen spezifischen Mikroprozessor-Bausteinsatz erfüllt werden können und welche Parameter für deren Auswahl maßgebend sind. Durch dieses Kapitel sollte klar geworden sein, auf welche Eigenschaften eines Mikroprozessors, sofern er für Datenübertragungs-Prozessoren angewendet wird, besonderes Augenmerk zu legen ist.

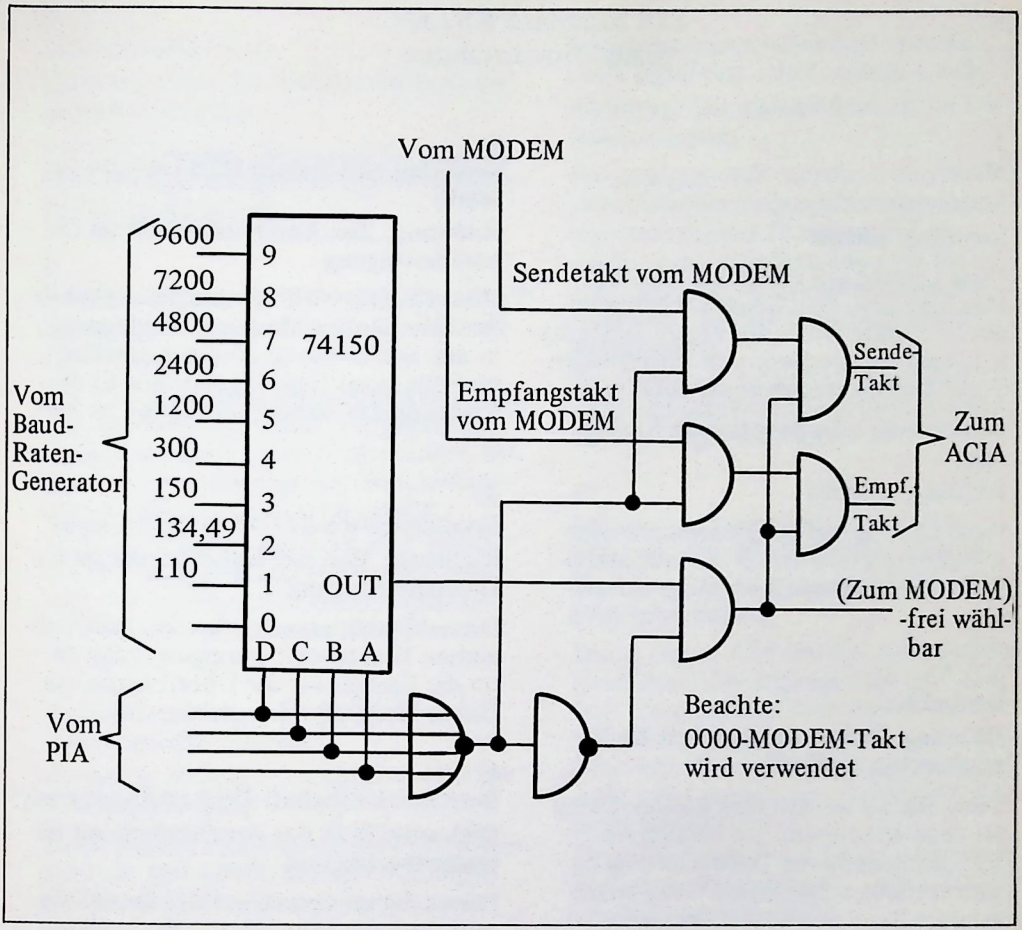


Bild 10.27
Programmierbarer Geschwindigkeitswähler

EIA Standard RS232C
Signal-Beschreibung

1.
Masse
Schutzerde
Richtung: entfällt

Diese Leitung soll galvanisch mit der Maschine bzw. dem Gehäuse verbunden sein. Sie sollte ferner mit externen Schutzleitungen entsprechend den zutreffenden Vorschriften zusammengeschaltet sein.

Betriebserde oder gemeinsame Rückleitung
Richtung: entfällt

Diese Leitung legt das gemeinsame Masse-Referenz-Potential für alle Stromkreise fest, ausgenommen die obige Schutzerde.

2.
Sendedaten
Richtung: Zur Anschlußeinheit für Datenübertragung (DAE)

Diese Signale werden vom lokalen Datenterminal erzeugt und der lokalen Anschlußeinheit für die Datenübertragung zum entfernt aufgestellten Datenterminal oder Zentralcomputer übergeben.

3.
Empfangsdaten
Richtung: Von der Datenendeinrichtung (DEE)

Diese Signale werden von der lokalen Anschlußeinheit für Datenübertragung erzeugt, wenn diese von der Anschlußeinheit für Datenübertragung des entfernt aufgestellten Datenterminals (DEE) Daten empfängt.

4.
Sendeteil einschalten RTS (request to send)
Richtung: Zur Anschlußeinheit für Datenübertragung

Dieses Signal wird verwendet, die lokale Anschlußeinheit für Datenübertragung in den Sende-Zustand zu bringen und im Falle eines Halb-Duplex-Kanals die Richtung der Datenübertragung zu steuern.

5.
Sendebereitschaft CTS (clear to send)
Richtung: Von der Anschlußeinheit für Datenübertragung

Dieses Signal, erzeugt von der Anschlußeinheit für Datenübertragung, zeigt an, ob die Einheit für die Übertragung von Daten bereit ist oder nicht.

6.
Betriebsbereitschaft DSR (data set ready)
Richtung: Von der Anschlußeinheit für Datenübertragung

Dieses Signal signalisiert die Betriebsbereitschaft der lokalen Anschlußeinheit für Datenübertragung.

7.
Datenendeinrichtung (DEE) betriebsbereit
Richtung: Zur Anschlußeinheit für Datenübertragung

Dieses Signal steuert die Aufschaltung der Anschlußeinheit für Datenübertragung auf die Anschlußleitung. Der EIN-Zustand bereitet die Anschlußeinheit für Datenübertragung für die Aufschaltung auf die Anschlußleitung vor und erhält die durch manuelle Hilfe erstellte Verbindung aufrecht (z.B. durch Handwahl).

8.
Ankommender Ruf
Richtung: Von der Anschlußeinheit für Datenübertragung

Der EIN-Zustand dieses Signals zeigt an, daß die Anschlußeinheit ankommenden Ruf empfangen hat.

9.
Empfangssignalpegel (received line signal)
Richtung: Von der Anschlußeinheit für Datenübertragung

Der EIN-Zustand dieses Signals zeigt an, daß die Anschlußeinheit für Datenübertragung ein Daten-Signal empfangen hat, das die vom Hersteller der Anschlußeinheit festgelegten Kriterien erfüllt.

Der AUS-Zustand zeigt an, daß entweder kein Daten-Signal empfangen wurde oder das empfangene Daten-Signal für die Demodulation ungeeignet ist.

10.
Ausreichender Empfangssignalpegel (Signal quality detector)
Richtung: Von der Anschlußeinheit für Datenübertragung

Dieses Signal zeigt an, ob die Wahrscheinlichkeit, fehlerhafte Daten empfangen zu haben, groß oder klein ist.

Der EIN-Zustand wird immer dann erzeugt, wenn es keinen Grund gibt, zu glauben, daß ein Fehler aufgetreten ist, d.h. der Empfangssignalpegel eine ausreichende Amplitude aufweist.

Der AUS-Zustand zeigt an, daß die Wahrscheinlichkeit für einen aufgetretenen Fehler hoch ist. In manchen Fällen veranlaßt dies gleichzeitig eine wiederholte Übertragung des unmittelbar vorher übertragenen Daten-Signals.
Vorzugsweise soll die Auswertung dieses Signals so ausgelegt sein, daß es eine Identifikation von eventuell fehlerhaften Signal-Elementen der empfangenen Daten erlaubt.

11.
Selektor für Datenübertragungsraten (data signal rate selector) DEE-Quelle
Richtung: Zur Anschlußeinheit für Datenübertragung

Dieses Signal wird benutzt zwischen den zwei Datenübertragungsraten im Falle von "synchronen Datensätzen" auszuwählen bzw. zwischen den zwei Bereichen von Datenübertragungsraten im Falle von "asynchronen Datensätzen" auszuwählen.

Der EIN-Zustand sollte der höheren Datenübertragungsrate bzw. dem höheren Bereich zugeordnet sein.

12.
Selektor für Datenübertragungsraten (data signal rate selector) DAE-Quelle
Richtung: Von der Anschlußeinheit für Datenübertragung

Dieses Signal wird benutzt zwischen den zwei Datenübertragungsraten im Falle von "synchronen Datensätzen" auszuwählen bzw. zwischen den zwei Bereichen von Datenübertragungsraten im Falle von "asynchronen Datensätzen" auszuwählen.

Der EIN-Zustand sollte der höheren Datenübertragungsrate bzw. dem höheren Bereich zugeordnet sein.

13.
Sendeschrittakt (transmitter signal element timing) DEE-Quelle
Richtung: Zur Anschlußeinheit für Datenübertragung

Dieses Signal versorgt die Anschlußeinheit für Datenübertragung mit Zeitinformation über den Sendetakt der Signal-Elemente.

Die negative Signalflanke (EIN → AUS) soll nominell die Mitte eines jeden Signal-Elementes der übertragenen Daten anzeigen.

14.
Sendeschrittakt (transmitter signal element timing) DAE-Quelle

Richtung: Von der Anschlußeinheit für Datenübertragung

Dieses Signal versorgt die Datenendeinrichtung (DEE) mit Zeitinformation über die Signal-Elemente. Die DEE soll Sendedaten bereitstellen, wenn eine positive Flanke (AUS → EIN) diesem Signal vorliegt.

15.
Empfangsschrittakt (receiver signal element timing) DAE-Quelle

Richtung: Von der Anschlußeinheit für Datenübertragung

Dieses Signal versorgt die DEE mit Zeitinformation über die empfangenen Signal-Elemente. Die negative Signalflanke (EIN → AUS) zeigt die zeitliche Mitte der empfangenen Signalelemente (Daten) an.

16.
Sendedaten-Kanal 2 (secondary transmitted data)

Richtung: Zur Anschlußeinheit für Datenübertragung

Diese Signale sind identisch mit den bereits beschriebenen Sendedaten, sie werden nur auf dem untergeordneten Kanal, Kanal 2, übertragen.

Die Signale werden vom Daten-Terminal (DEE) erzeugt und der zweiten lokalen Anschlußeinheit für Datenübertragung zur Übertragung an das entfernt aufgestellte DEE zugeführt.

17.
Empfangsdaten-Kanal 2 (secondary received data)

Richtung: Von der Anschlußeinheit für Datenübertragung

Diese Signale sind identisch mit den bereits beschriebenen Empfangsdaten, sie werden nur auf dem untergeordneten Kanal, Kanal 2, empfangen.

18.
Sendeteil Kanal 2 einschalten (secondary request to send)

Richtung: Zur Anschlußeinheit für Datenübertragung

Dieses Signal ist identisch mit "Sendeteil einschalten", es wählt nur den Kanal 2 anstelle des Hauptkanals für die Übertragung an.

19.
Sendebereitschaft-Kanal 2 (secondary clear to send)

Richtung: Von der Anschlußeinheit für Datenübertragung

Dieses Signal ist identisch mit "Sendebereitschaft", es zeigt nur die Verfügbarkeit des Kanals 2 anstelle des Hauptkanals an. Dieses Signal wird nicht bereitgestellt, wenn Kanal 2 nur zur Stromkreis-Absicherung oder als Interruptkanal ausgeführt ist.

20.
Empfangssignalpegel-Kanal 2 (secondary received line signal detector)

Richtung: Von der Anschlußeinheit für Datenübertragung

Dieses Signal ist identisch mit 9., es zeigt nur den richtigen Empfang des Leitungssignals von Kanal 2 an.

KAPITEL 11

ANWENDUNGSGEBIETE FÜR MIKROPROZESSOREN: VERKAUFSSTELLEN-TERMINALS

Dieses Kapitel definiert die Anforderungen an ein Kassen- bzw. Verkaufsstellen-Terminal und beschreibt ein spezifisches, mit einem Mikroprozessor aufgebautes Terminal.

Anwendungsgebiete für Mikroprozessoren: Verkaufsstellen-Terminals

Dieses Kapitel definiert das Verkaufsstellen- bzw. POS- (point-of-sale) Terminal, das im allgemeinen als elektronische Registrierkasse bekannt ist. Die Diskussion wird mit allgemein erforderlichen Funktionen eingeleitet, unterscheidet im weiteren Verlauf verschiedene Methoden der Ausführung dieses Terminal-Typs und beschreibt hierzu ein spezifisches Beispiel.

Allgemeine Merkmale eines Verkaufsstellen-Terminals
(Bild 11.1)

Im wesentlichen handelt es sich bei einem Verkaufsstellen-Terminal (POS) um eine elektronische Registrierkasse, die, unter Einsatz der Leistung eines Mikroprozessors, die Funktionen eines modernen Buchungs- bzw. Abrechnungssystems übernimmt. Das Verkaufsstellen-Terminal besitzt im allgemeinen sämtliche bzw. die meisten der in Bild 11.1 dargestellten Leistungsmerkmale.

Fähigkeiten eines Verkaufsstellen-Terminals
(Bild 11.2)

Bild 11.2 zeigt eine Aufstellung, welche die Leistungsfähigkeit eines Verkaufsstellen-Terminals veranschaulicht. Mit Mikroprozessoren aufgebaute Terminals addieren Preise, berechnen die Geldrückgabe (Wechselgeld), drucken Kunden- sowie Protokollbelege aus wie eine normale Registrierkasse. Darüber hinaus ist aufgrund der Prüfung einer Kreditkarte bzw. einer Kontonummer und Durchführung eines Vergleiches mit einer lokalen Datenbank oder mit einem Hauptrechner eine Aufrechnung der neuesten Kreditinformation augenblicklich abrufbar. Durch Eingabe eines Produktcodes und der Menge bei jeder Transaktion kann eine Echtzeit-Inventur erstellt werden. Die jeweilige Mehrwertsteuer wird automatisch durch eine Nachschlagtabelle berechnet, wodurch Bedienfehler reduziert werden. Auch kann das Terminal dahingehend programmiert werden, die Bedienperson zur Korrekturvornahme aufzufordern, wenn ungültige oder unzutreffende Informationen eingegeben wurden. Das Terminal kann zu jeder Form der Aufzeichnung eingesetzt werden, und zwar über Magnetbandkassetten oder ähnliche Massenspeicher oder auch nur mit der Möglichkeit des Bandausdrucks, gekoppelt mit dem internen Speicher des Mikroprozessors. Die Daten zur Arbeitszeiterfassung können über das Bedienfeld, oder andere Eingabe-Einheiten, in das Terminal eingegeben werden, um somit Informationen für eine Lohn- und Gehaltsabrechnung zu liefern.

Management-Informationspanel bei Verkaufsstellen-Terminal
(Bild 11.3)

Da das Verkaufsstellen-Terminal Auskunft über den Verkauf, Kredite, Lagerbestand, Kassenbestand und Lohn- und Gehaltsregister gibt, kann es als kleines, kompaktes und preiswertes Management-Informationssystem aufgefaßt werden, wie es in Bild 11.3 angedeutet ist.

- Standard-Registrierkassenfunktionen
- Kreditprüfung
- Inventarkontrolle
- Steuerberechnungen
- Fehlererlernung und -korrektur
- Aufstellung von Lohn- und Gehaltslisten
- Karteiverwaltung

Bild 11.1
Allgemeine Merkmale eines Verkaufsstellen-Terminals

- Aufrechnung von Artikelpreisen
- Gesamtpreisangabe
- Berechnung von Wechselgeld
- Ausdrucken von Kundenrechnungen
- Ausdrucken eines Protokollbelegs
- Kreditprüfung über Kontonummer
- Laufende Lagerbestandskontrolle
- Automatische Mehrwertsteuer-berechnung
- Anzeige erforderlicher Korrektur-vorgänge bei Fehlern der Bedien-person
- Vollständige Karteiführung
- Bestandsdatei für Personaldaten

Bild 11.2
Fähigkeiten eines Verkaufsstellen-Terminals

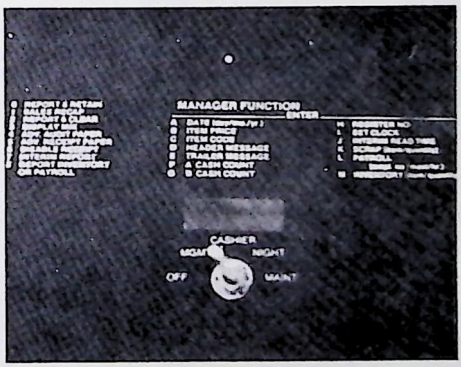


Bild 11.3
Management-Information-Paneel-Verkaufs-stellen-Terminals

Terminal-Typen
(Bild 11.4)

Zur Implementierung von Verkaufsstellen-Systemen werden im wesentlichen drei verschiedene Methoden verwendet. Diese sind näher in Bild 11.4 aufgelistet. Der Hauptunterschied liegt im Intelligenzgrad des Terminals. Äußerlich betrachtet mag dem Benutzer sowie dem Verkäufer die Leistung dieser verschiedenen Systeme gleich erscheinen, jedoch ist insgesamt der Betrieb unterschiedlich.

Umspeicher-Terminal
(Bild 11.5)

Ein über einen Zentralrechner gesteuertes Umspeicher-Terminal (dumb terminal) ist in Bild 11.5 dargestellt. Die Bezeichnung "Umspeicher-Terminal" rührt daher, daß dieses nicht unabhängig arbeiten kann, sondern lediglich mit jenen Daten und Operationen, die von einem zentral angeordneten Rechner über bestimmte Übertragungskanäle übermittelt werden. Diese Terminals sind im wesentlichen Übertragungs-Steuergeräte mit geringem oder keinem lokalen Daten- oder Programmspeicher.

Hauptrechner-Verkaufsstellennetz (Abfrageterminals)
(Bild 11.6)

Das Umspeicher-Terminal gibt dem Benutzer effektiv größere Möglichkeiten bei gleichzeitiger Senkung der Terminalkosten. Bild 11.6 zeigt ein Multi-Terminalsystem mit dem Zentralcomputer, der jedes Terminal abfragt. Einer der größten Nachteile des Umspeicher-Terminals ist seine Zuverlässigkeit: setzt der Zentralrechner aus, so werden gleichfalls sämtliche Terminals außer Betrieb gesetzt. Auch ist das Terminal anfällig gegenüber jeglichen Ausfällen im Übertragungssystem, das die Verbindung mit dem Zentralrechner herstellt. Das Übertragungssystem wirft nicht nur Probleme in Bezug auf Zuverlässigkeit auf, sondern es kann auch ein wesentlicher Kostenfaktor für das Gesamtsystem darstellen.

Die Terminals werden nicht konstant vom Hauptrechner bedient, sondern lediglich zyklisch abgefragt. Hierdurch werden die Leitungskosten sehr stark herabgesetzt, aber die Ansprechzeit des Terminals wird länger. Würden die Terminals nicht abgefragt, so wäre jeweils eine Leitung vom Hauptrechner zu jedem Terminal erforderlich, was die Kosten stark erhöht, aber nur eine kleine Verbesserung der Leistungsfähigkeit ergibt.

Hauptrecher-Verkaufsstellennetz (Fernsteuereinheiten)
(Bild 11.7)

Eine Möglichkeit, die Anzahl der Verbindungsleitungen zu reduzieren und dennoch eine gute Ansprechzeit zu dem Umspeicher-Kassenterminal beizubehalten, besteht darin, die Intelligenz in Satelliten-Prozessoren oder Steuereinheiten unterzubringen (siehe Bild 11.7). Dieses Verfahren arbeitet mit dem Umspeicher-Terminal, es wird jedoch die Intelligenz aus dem Zentralrechner herausgenommen und in Satelliten-Prozessoren oder Haupt-POS-Stationen untergebracht. Dieses System kann mit einem Zentralrechner verbunden werden, hängt jedoch betrieblich nicht mehr von letzterem ab. Auch wächst hierdurch die Zuverlässigkeit gegenüber dem System nach Bild 11.6, und ein Ausfall des Hauptterminal-Prozessors kann nur das gesamte lokale Netzwerk oder eine Anzahl von Terminals stillegen. Der Haupt-Terminal-Prozessor wirkt als Übertragungsprozessor, wenn dieser mit einem Zentralrechner verbunden ist und liefert darüber hinaus Intelligenz an die Verkaufsstellen-Unterterminals.

- Umspeicher-Terminal – vom Hauptrechner gesteuert
- Umspeicher-Terminal – Fernsteuerung
- Unabhängiges Terminal – kleines Terminal

Bild 11.4
Terminal-Typen

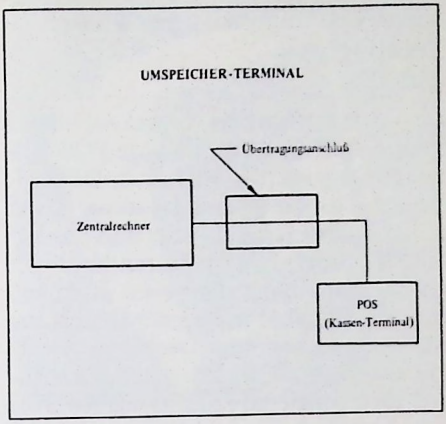


Bild 11.5
Umspeicher-Terminal

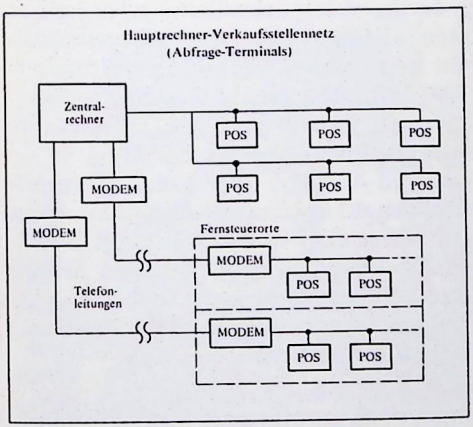


Bild 11.6
Hauptrechner-Verkaufsstellennetz (Abfrage-Terminals)

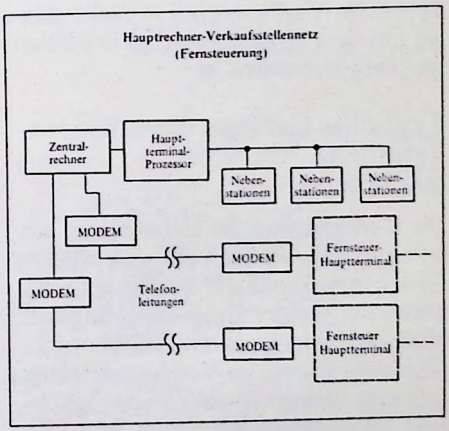


Bild 11.7
Hauptrechner-Verkaufsstellennetz (Fernsteuerung)

Netzwerk unabhängiger Kassen-Terminals
(Bild 11.8)

Die dritte Kategorie der Verkaufsstellen-Terminals ist das unabhängige POS-Terminal, dargestellt in Bild 11.8. Dieses Terminal umfaßt einen Prozessor sowie die erforderlichen Speicher und das Speicherprogramm zur Durchführung eines unabhängigen Betriebes. Es kann über ein MODEM mit einem Hauptrechner verbunden werden, wobei das Terminal auch noch als Übertragungs-Steuereinheit fungieren muß. Diese Konfiguration wird als die zuverlässigste bezeichnet, da der Ausfall irgendeines Systemteils auf das betroffene Teil begrenzt bleibt. Allgemein gesehen, ist diese Form kostspieliger und es stehen nicht die gleichen Datenmengen wie bei den anderen Typen zur Verfügung, da es nicht ständig mit dem großen Speicher des Zentralrechners verbunden ist.

Typisches Gehäuse eines Kassen-Terminals
(Bild 11.9)

Die Konstruktion des Gehäuses ist aus humanitären Gründen ein wesentlicher Gesichtspunkt, da das Terminal im allgemeinen in einer Umgebung eingesetzt wird, die einen längeren Dauerbetrieb, eine hohe Dichte an Verkaufsvorgängen und eine geringe Ausbildungsmöglichkeit des Bedienenden bietet. Bild 11.9 zeigt ein typisches Verkaufsstellen-Terminal.

Frontplatte eines Kassen-Terminals
(Bild 11.10)

Eine der grundsätzlichen Konstruktionsbetrachtungen beim POS-Terminal ist die Bedienungskonsole, welche Bild 10.10 zeigt. Für die meisten Industriezweige liegen genau definierte Normen vor, die eine maximale Effektivität der Bedienungsperson, bei geringster Ermüdung, sicherstellen. Über die Auslegung der Bedienungskonsole hinaus sind weitere menschlich bedingte Faktoren zu berücksichtigen, wie z.B. einfacher Wechsel der Bänder, Zugänglichkeit des Druckers, einfache Möglichkeiten der vorbeugenden Wartung sowie die Möglichkeit für eine ungelernte Bedienungsperson die Beschriftung und die Funktion der einzelnen Tasten einfach zu verstehen.

Blockdiagramm eines unabhängigen Kassen-Terminals
(Bild 11.11)

Bild 11.11 zeigt ein Blockdiagramm eines typischen unabhängigen POS-Terminals. Die Telefonleitung sowie das MODEM werden eingesetzt, sofern eine Verbindung mit dem Zentralrechner besteht. Ein POS-Terminal dieser Art muß in der Lage sein, sämtliche erforderlichen arithmetischen und Steuerfunktionen auszuführen. Aus diesem Grunde wird dieses mit einem Mikroprozessor mit zusätzlichem Speicher, sofern erforderlich, ausgerüstet. Ferner muß die Ein- und Ausgabe über Tastatur, Anzeigefeld und Drucker gegeben sein. Außerdem sollte eine typische Einheit über verschiedene Hilfseinrichtungen für den Benutzer verfügen, wie z.B. Tastensperre, Überwachungsanzeigen, Alarmvorrichtungen, Registrieranzeige usw.

Zur näheren Beschreibung eines typischen POS-Terminals befaßt sich dieses Kapitel mit Konstruktion und Ausführung eines Schnellrestaurant-Terminals für den Einsatz in einem beliebigen Schnellrestaurants mit Straßenverkauf. Das Register hat die Preise der verkauften Waren zu addieren. Da es sich hierbei um eine begrenzte Auswahl handelt, geht die Konstruktion davon aus, für die einzelnen Waren Druckknöpfe einzusetzen, die jeweils für eine bestimmte Ware codiert sind. Mengentasten werden eingesetzt, um die Anzahl der gleichen Artikel zu erfassen.

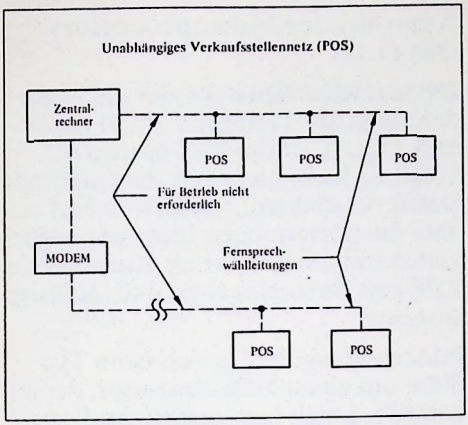


Bild 11.8
Netzwerk unabhängiger Kassen-Terminals

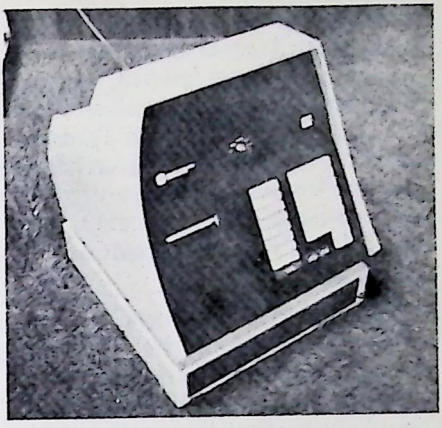


Bild 11.9
Typisches Chassis eines Kassen-Terminals

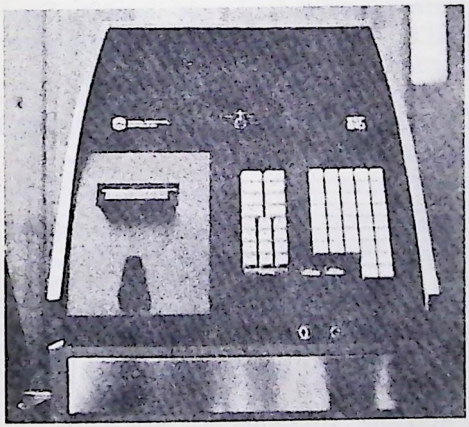


Bild 11.10
Frontplatte des Verkaufsstellen-Terminals

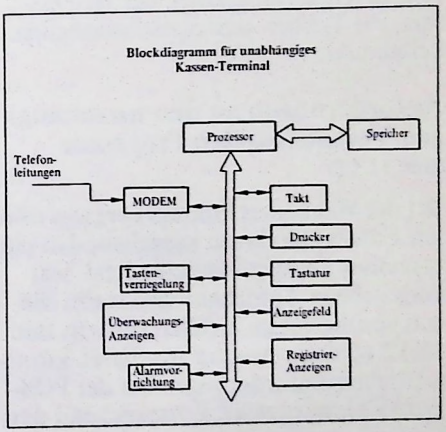


Bild 11.11
Blockdiagramm eines unabhängigen Verkaufs-stellen-Terminals (POS)

Darüber hinaus ist ein Anzeigefeld für die einzelnen Transaktionen für zwei Druckerausgänge vorzusehen, und zwar einen für den Kunden und einen für ein Protokollband. Während der Transaktion werden außerdem noch verschiedene andere Vorkehrungen getroffen, die z.B. Paneelanweisungen für die richtige Betriebsfolge, Fehlerkorrektur usw. einschließen.

Das Herz des POS-Terminals ist sein Mikroprozessor. Da sämtliche Vorgänge im Terminal durch den Mikroprozessor gesteuert werden, ist seine Wahl eine kritische Entscheidung für den Entwickler. Bei der Auswahl des Mikroprozessors sind verschiedene Faktoren zu berücksichtigen. Diese schließen die Anzahl der Eingabe/Ausgabe-Ereignisse pro Zeiteinheit ein, ferner Interruptanforderungen, Befehlszykluszeiten, der Befehlssatz, die Treiber- und Speichermöglichkeiten usw.

Anforderungen an den unabhängigen Verkaufsstellen-Prozessor
(Bild 11.12)

Bei der Wahl eines Mikroprozessors muß der Entwickler davon ausgehen, daß ein typisches Terminal verschiedene, fest vorgegebene Anforderungen stellt, die von vornherein zu erfüllen sind. In Bild 11.12 wird ein Auszug aus diesen grundsätzlichen Anforderungen bei der POS-Entwicklung gezeigt. Entsprechend der Darstellung liegt die maximale Datengeschwindigkeit bei etwa 1200 Ereignissen pro Sekunde, die ohne weiteres von den meisten integrierten Mikroprozessoren bewältigt werden kann. Darüber hinaus verfügen diese Mikroprozessoren auch über sämtliche arithmetische und logische Fähigkeiten zur einfachen Bearbeitung der erforderlichen mathematischen Vorgänge der Registrierkasse. Das in diesem Beispiel behandelte POS-Terminal erfordert sieben Interrupts, vier Eingangs- und zehn Ausgangsadressen.

Auswahl des Mikroprozessors
(Bild 11.13)

Ein spezieller Baustein, der die Anforderungen des Terminals erfüllt (siehe Bild 11.13), ist der Mikroprozessor 8008. Er kann nicht nur die anfallenden Daten verarbeiten, sondern verfügt auch über die notwendigen Interruptmöglichkeiten mit einer mehr als ausreichenden Zahl von Dateneingangs- und -ausgangsadressen.

Prinzipiell handelt es sich beim Typ 8008 um einen MOS-Prozessor, der mit der TTL-Logik kompatibel ist. Es besteht die Möglichkeit, Speicher mit einer Kapazität bis zu 16 KByte direkt anzuschließen. Die Zentraleinheit (CPU) enthält eine parallel arbeitende 8-Bit-ALU (arithmetische, logische Einheit), ein dynamisches RAM mit sieben 8-Bit-Datenregistern und einen 8x4-Bit-Stack mit kompletter Befehlsdecodierung und Steuerlogik.

Systembaugruppen eines Verkaufsstellen-Terminals
(Bild 11.14)

Nach erfolgter Auswahl des Mikroprozessors besteht der nächste Schritt darin, jede der vom Mikroprozessor gesteuerten Einheiten zu prüfen. Wie in Bild 11.14 dargestellt, kann ein POS-System in drei Baugruppen untergliedert werden. Diese sind „Eingabe", „Ausgabe" und eine breitgestreute Gruppe, die mit „interne Funktionen" bezeichnet werden kann.

Ereignis	Ereignisse/s	Interrupt	Eingabe	Ausgabe
• Übertragungen	1200	Ja (2 erforderl.)	Ja + Status	Ja
• Druckerquittung	30-100 Zeichen oder 2 Zeilen	Ja	Status	Ja
• Protokolldrucker	30-100 Zeichen oder 2 Zeilen	Ja	Status	Ja
• Anzeige	–	Nein	keine	Ja
• Tastatur	20-40 (1000 Trenn.)	Ja	Ja	keine
• Überwach.-Paneel	–	Nein	keine	Ja
• Taktgeber	–	Nein	Ja	Ja
• Registrieranzeige	–	Nein	Status	Ja
• Netz Ein/Aus	–	Ja	keine	keine
• Alarmanzeige	–	Nein	keine	Ja
• Tastenverriegl.	–	Nein	Status	Ja
• Wartung	–	Ja	Ja	Ja

Bild 11.12
Anforderungen an den unabhängigen Verkaufsstellen-Prozessor

Funktion	erforderliche Anzahl	8008
Interrupts	7	8
Eingangs-adressen	4	8
Ausgangs-adressen	10	24

Bild 11.13
Mikroprozessor-Auswahl

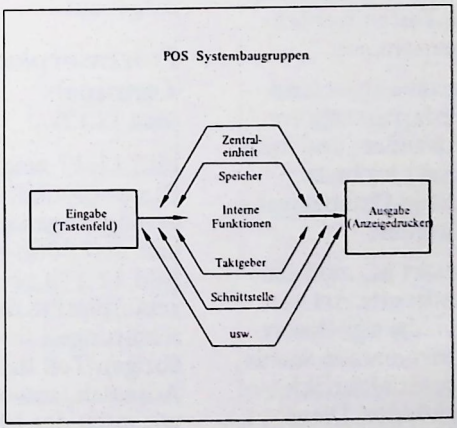


Bild 11.14
Systembaugruppen eines Kassen-Terminals

Konstruktionsfaktoren für das Tastenfeld
(Bild 11.15)

Das Tastenfeld ist die primäre Eingabeeinheit für dieses Schnellrestaurant-Terminal. Bei der Konstruktion des Tastenfeldes ist eine Reihe von Faktoren zu berücksichtigen, die in Bild 11.15 zusammengefaßt sind.

Durch die Eigenschaft des "Roll-Over" können mehr als eine Taste nacheinander gedrückt werden, ohne daß ein Datenverlust oder die Übernahme von zusätzlichen fehlerhaften Daten erfolgt. Einige Tastaturen arbeiten mit einem sogenannten "2-Key-Roll-Over", das die folgende Reihenfolge ermöglicht: zuerst wird Taste A gedrückt, dann wird Taste B gedrückt, und beide Tasten bleiben gedrückt, dann wird die erste Taste A freigegeben, dann wird B freigegeben. Die Daten von beiden Tasten werden somit einwandfrei übernommen.

Bezüglich der Federspannung ist eine Feder mit positiver Federwirkung von etwa 2 N Kraft zu verwenden, und zwar im Gegensatz zu den sehr leicht zu drückenden Tasten, die in Dateneingabesystemen verwendet werden.

Wenn eine Taste gedrückt ist, muß die Bedienperson eine bestimmte Art von Rückmeldung erhalten, die signalisiert, ob die Information übernommen wurde, da die Bedienperson sonst Mehrfacheingaben verursachen könnte. Diese Rückmeldung erfolgt durch einen kleinen Lautsprecher, der aus einem kurzen 1-kHz-Rechtecksignal ein hörbares Klickgeräusch erzeugt, das den Klang des Tastenfeldes mehr einer mechanischen Registrierkasse annähert, die den meisten Benutzern vertraut ist.

Ein bei einem Tastenfeld typisches Problem ist der sogenannte Stoßbetrieb. Eine Bedienperson kann mehreren Eingaben mit der gleichen oder mit benachbarten Tasten schneller eingeben, als ein Prozessor in der Lage ist, diese zu verarbeiten, was schließlich zu einem Verlust an Daten führt. Dieses Problem kann durch Zwischenschaltung eines Silo-Speichers (FIFO = first in/first out) gelöst werden, der zwischen Tastenfeld und Prozessor als Zwischenspeicher dient. Er nimmt bei den von der Tastatur ausgelösten Stoßbetriebsphasen die eingegebenen Daten auf und gibt sie in der richtigen Reihenfolge mit der Verarbeitungsgeschwindigkeit des Prozessors wieder ab.

Logikplatinen des Kassen-Terminals
(Bild 11.16)

Als nächstes muß der Betriebsbereich betrachtet werden. Dieses bestimmte POS-Terminal wurde mit den sechs in Bild 11.16 angegebenen Logikplatinen aufgebaut.

Prozessorplatine des Kassen-Terminals
(Bild 11.17)

Bild 11.17 zeigt die Prozessorplatine. Dieses Blockdiagramm der als Beispiel herangezogenen Prozessorplatine enthält den 8008-Prozessorchip.
Bild 11.17a zeigt, daß sich auf der Prozessorplatine die Takt- und Steuerschaltungen für den Prozessor und den übrigen Teil des Systems befinden. Adressen, sowohl die des Speichers wie auch der Ein-/Ausgabe, sowie die Daten werden in zwei Latches gespeichert. Bedingt durch die begrenzte Anzahl der Gehäuseanschlüsse des 8008 werden diese Informationen während zwei Unterzyklen vom 8008 ausgegeben.

Die Interrupt-Hardwareschaltung ist in Bild 11.17b gezeigt. Diese besteht aus einem Prioritätscodierer, einem Latch und einem Decodierer, der einen auftretenden Interrupt als erkannt quittiert. Die Aufgabe dieses Interruptsystems besteht darin, den Prozessor zur

- Tasten "Roll-Over"
- Federspannung
- Rückmeldung für Bedienperson
- Tastenanordnung
- Stoßbetrieb

Bild 11.15
Konstruktionsfaktoren des Tastenfeldes

- Prozessorplatine
- Speicherplatte
- Allgemeine Schnittstellenplatine
- Asynchrone Datenübertragungseinheit
- MODEM
- Drucker-Schnittstelle

Bild 11.16
Logikplatinen eines Kassen-Terminals

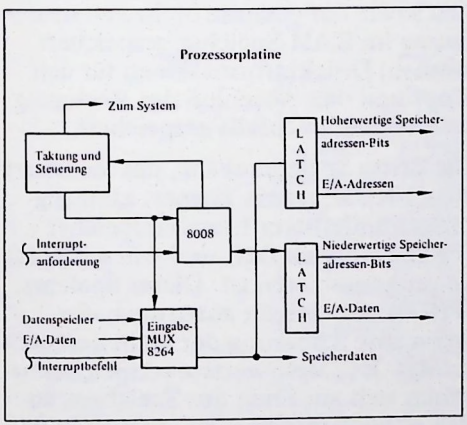
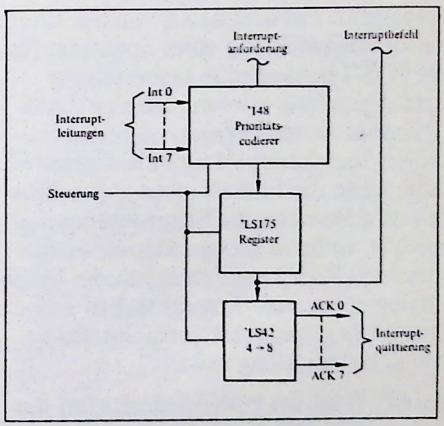


Bild 11.17 (a)
Prozessorplatine eines Kassen-Terminals



(b)

Ausführung eines eingegebenen Befehls zu zwingen, und zwar während des Interruptabtastzyklus des Prozessors. Dieser Befehl ist effektiv ein Sprung zu einem Unterprogramm. Dieses Verfahren liefert dem Prozessor ein Prioritätsvektor-Interruptsystem, was insofern von Bedeutung ist, als es den schnellen Echtzeit-Eingabe- und Übertragungsfunktionen eine hohe Priorität zuordnet. Deshalb werden diese zuerst bearbeitet, ohne daß dabei Zeit durch ein Programm verloren geht, das feststellt, welcher Interrupt als nächster bearbeitet werden sollte.

Speicherplatine des Kassen-Terminals
(Bild 11.18)

Ein typisches Schnellrestaurant-POS-Terminal arbeitet, wie in Bild 11.18 dargestellt, mit drei Arten von Speichern. Bei der Entwicklung eines Speichers für ein POS-Terminal muß insbesondere darauf geachtet werden, daß der RAM (Random-Access-Memory-)Speicher seinen Speicherinhalt nicht verlieren darf, wenn die Einheit nicht angeschlossen ist oder wenn die Stromversorgung ausfällt. Denn in diesem Schreib-/Lesespeicher (RAM) werden sämtliche Informationen von den Transaktionen des Terminals gespeichert – Salden, Inventuren, Buchhaltung usw..

Bei der Wahl des POS-Speichers hat der Entwickler zwei Möglichkeiten: den Einsatz eines nichtflüchtigen Speichers (Magnetkernspeicher) oder eines flüchtigen Halbleiterspeichers (MOS-RAM) mit einer Batteriepufferung.

Organisation des Speichers
(Bild 11.19)

Die gesamte Organisation des Speichers ist in Bild 11.19 dargestellt. Das auszuführende Programm ist in einem maskenprogrammierten ROM gespeichert. Dieses Programm ist die Betriebs-Software für das Terminal und wird mitunter als Firmware bezeichnet. Dieser Festwertspeicher (ROM) ist ein nichtflüchtiger Speicher und befindet sich im unteren Teil des Speichers. Ein großer Vorteil dieser Anordnung besteht darin, daß mit der gleichen Hardware durch Austausch des Programm-ROMs die Betriebscharakteristiken der Maschine verändert werden können.

Der nächste Teil ist ein Schreib-/Lesespeicher (RAM). Hier werden die sich ändernden Informationen gespeichert, wie zum Beispiel Preise, sowie die drei Ziffern umfassende Konstante für die Konstanten-Tasten des Tastenfeldes. Der Verkaufsstellen-Leiter kann somit die Preise in einem POS-Terminal über das Tastenfeld in einigen Sekunden ändern. Darüber hinaus können sämtliche Inventur- und Buchhaltungsinformationen sowie der gesamte Mehrwertsteuerbetrag im RAM-Speicher gespeichert werden. Druckinformationen für den Kopf und den Abschluß der Rechnung werden hier ebenfalls gespeichert.

Die dritte Speicherebene, das Mehrwertsteuer-ROM, ist ein kleiner, einmalig programmierbarer Festwertspeicher (PROM), der einfach und wirtschaftlich zu programmieren ist. Dieser Speicher (PROM) ist einfach auszuwechseln, wenn eine Änderung der Mehrwertsteuer erfolgt. Der Mehrwertsteuerspeicher befindet sich am Ende des Speichers, so daß es bedeutungslos ist, wieviel ROM oder RAM im unteren Teil des Speichers eingesetzt werden, da das Programm in jedem Falle weiß, wo die Steuertabellen gespeichert sind.

Allgemeine Schnittstellenplatine des POS-Terminals
(Bild 11.20)

Die allgemeine Schnittstellenplatine in Bild 11.20 verbindet das Tastenfeld, die numerische Anzeige, den Antrieb der Kassenlade, die Alarmvorrichtung,

- Programm-ROM (Festwertspeicher)
- Datenspeicher-RAM (Schreib/Lesespeicher)
- Mehrwertsteuer-PROM (programmierbarer Festwertspeicher)

Bild 11.18
Verkaufsstellen-Speicherplatine

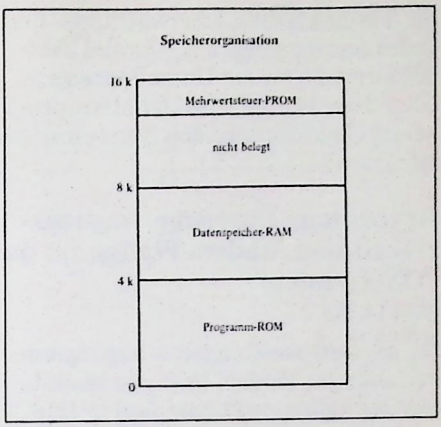


Bild 11.19
Organisation des Speichers

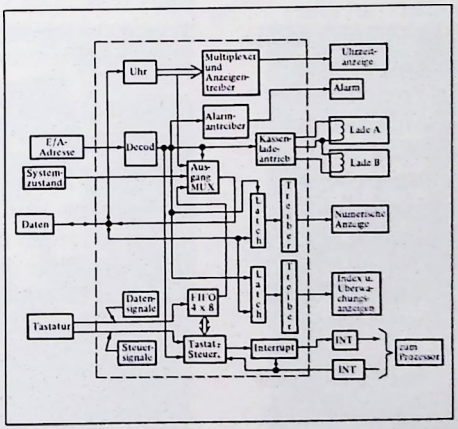


Bild 11.20
Allgemeine Schnittstellenplatine

die Tastenschalter, Überwachungs- und Indexanzeige sowie die Uhr und die Schnittstelle für die Uhrzeitanzeige. Über diese Platine wird der Mikroprozessor ebenfalls über den Systemzustand informiert.

Asynchrone Datenübertragungseinheit und Modem-Platine für das POS-Terminal
(Bild 11.21)

Die asynchrone Datenübertragungseinheit und das Modem bzw. die Modulator/Demodulator-Platine sind in Bild 11.21 dargestellt. Sie ermöglichen dem Terminal die Verbindung zu einem Zentralrechner über Telefonleitungen. Die Betriebsweisen der beiden Platinen wurden im einzelnen in Kapitel 10 erläutert.

Werden Informationen vom Zentralrechner empfangen, gewöhnlich mit einer Geschwindigkeit von 1200 Bits/s (Baud), so unterbricht die asynchrone Datenübertragungseinheit den Prozessor. Der Prozessor liest anschließend das empfangene Wort ein und verarbeitet es. Um ein Wort zu übertragen, sendet der Prozessor ein Wort an die asynchrone Datenübertragungseinheit, die den Prozessor nach Senden des Wortes unterbricht; anschließend ist er für ein neues Wort bereit.

Drucker-Schnittstellen-Platine des POS-Terminals
(Bild 11.22)

Die letzte Logikplatine dieses Systems ist die Drucker-Schnittstelle, wie in Bild 11.22 dargestellt. Normalerweise befinden sich zwei Druckstreifen im Terminal, eines für den Kunden und eines als Protokollbeleg. Die Schnittstellenplatine treibt dann beide Drucker und bildet eine durch Interrupt-gesteuerte Einheit mit voller Zeilenpufferung. Die Drucker-Schnittstelle arbeitet mit einem gemeinsamen Puffer für beide Drucker, der vom Prozessor geladen wird. Der Druckbefehl, ein Eingabe/Ausgabe-Befehl des Prozessors, definiert, welchem Drucker der Puffer zugeordnet wird. Die Schnittstelle arbeitet anschließend als ein Steuernetzwerk, das den Pufferinhalt zum richtigen Zeitpunkt zum Drucker transportiert. Unter Verwendung dieses Verfahrens ist es nicht erforderlich, daß die Schnittstelle doppelt ausgeführte Schaltungen besitzt. Sie wäre dann erforderlich, wenn eine getrennte Schnittstelle für jeden Drucker eingesetzt würde.

Typisches Kassen-Terminal – ECR 810
(Bild 11.23)

Bild 11.23 zeigt ein typisches POS-Terminal. Hierbei handelt es sich um das vollständige ECR 810-Terminal, das von Digital Computer Controls hergestellt wird. Die ECR 810 ist ein unabhängiges Kassen-Terminal, das speziell für Schnellrestaurants entwickelt wurde.

ECR 810-Tastatur
(Bild 11.24)

Wie in Bild 11.24 dargestellt, umfaßt die Eingabe eine Tastatur mit einer Anzahl von festen Artikeltasten, die einen Preis für jeden Artikel und eine drei Buchstaben umfassende Kurzbezeichung, die auf der Rechnung ausgedruckt wird, erzeugen.

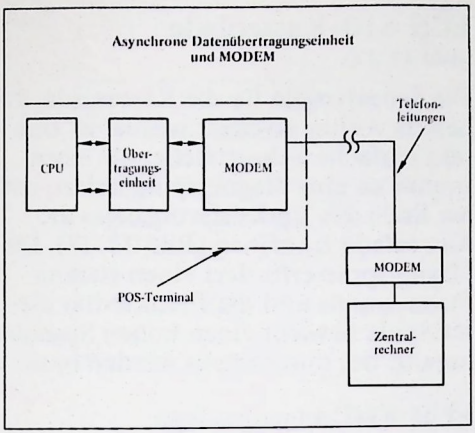


Bild 11.21
Asynchrone Datenübertragungseinheit und Modemplatine für das POS-Terminal

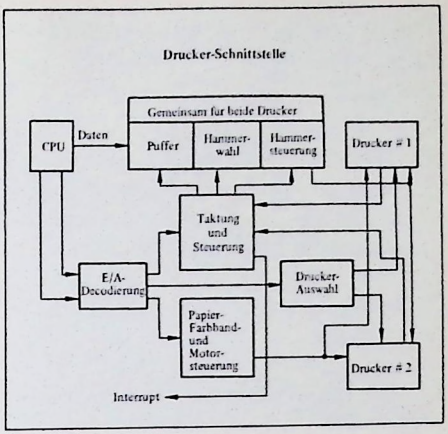


Bild 11.22
Drucker-Schnittstellenplatine für das POS-Terminal

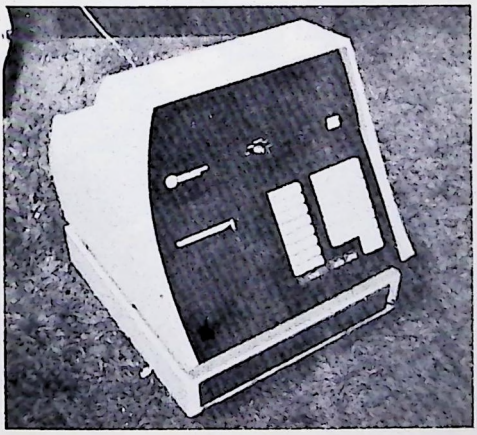


Bild 11.23
Typisches Kassen-Terminal – ECR 810

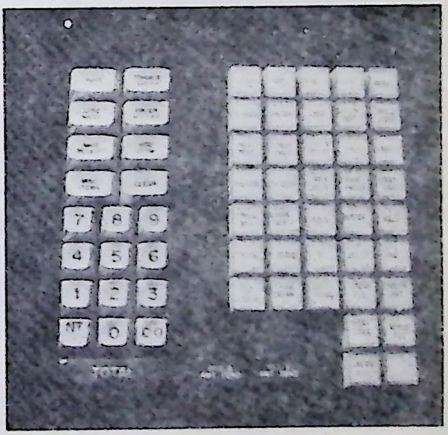


Bild 11.24
ECR 810-Tastatur

Typischer Rechnungs-Coupon des ECR 810
(Bild 11.25)

Wenn die Tasten gedrückt sind, so zeigt das Terminal den vorgegebenen Preis für diesen Artikel an, und, nach der Zusammenrechnung, wird der Preis ausgedruckt sowie eine drei Buchstaben umfassende Kurzbeschreibung des Artikels auf dem Rechnungs-Coupon, wie in Bild 11.25 dargestellt.

ECR 810-Anzeige
(Bild 11.26)

Die verwendete numerische Anzeige ist ein großes 4stelliges Anzeigefeld mit Kaltkatoden-Röhren, die gut lesbar und wirtschaftlich ist. Obwohl diese Anzeigeelemente auf ihre Größe bezogen preisgünstig sind, erfordern sie jedoch 200 Volt, was die Netzteilkosten erhöht. Die Ansteuerung dieser Anzeigenelemente ist jedoch kein Problem, da einfache integrierte Decodiertreiberschaltungen erhältlich sind.

Die Anzeigeeinheit ist so aufgebaut, daß die Aufnahme einer Echtzeituhr möglich ist. Die Uhr wird für die Arbeitszeiterfassung eingesetzt, aber auch für die automatische Registrierabfrage zu vorher bestimmten Zeitpunkten. Die Uhrzeiterfassung wird nicht durch Interrupts gesteuert, da es sich hier nicht um Echtzeit- bzw. zeitkritische Vorgänge handelt. In Zeiten, in denen sich der Prozessor im Wartezustand befindet und keine Verarbeitung erforderlich ist, liest er kontinuierlich die Uhrzeit, um festzustellen, ob ein vorher bestimmter Zeitpunkt erreicht ist. Auf diese Weise wird der Prozessor nicht durch Interrupts von der Uhr während kritischen Operationen unterbrochen.

ECR 810-Kassenlade
(Bild 11.27)

Die Schnittstelle für die Kassenlade, die bereits vorher erwähnt wurde, ist eine sehr einfache Schnittstelle, die einen Impuls an eine Magnetspule liefert, um am Ende des Verkaufsvorganges die Kassenlade zu öffnen (Bild 11.27). Die Magnetspule erfordert einen starken Stromimpuls und die Induktivität dieser Spule bewirkt einen hohen Spannungsimpuls, der unterdrückt werden muß.

ECR 810-Alarmanlage
(Bild 11.28)

Die Alarmanlage ist ein einfacher kleiner Lautsprecher, wie in Bild 11.28 dargestellt, der entweder zur Rückmeldung ein Klick-Geräusch immer dann abgibt, wenn eine Taste gedrückt wird oder der einen hohen Ton aussendet, um die Bedienperson auf einen fehlerhaften Zustand hinzuweisen.

12. Dez. 74 mit besten Wünschen

2 FSH	3.78	BBQ	2.59
RIB	3.69	CBG	1.19
SHR	1.99		
Mehrwert-steuer	.66	Gesamt	13.90
verein-nahmt	15.00	Wechselgeld	1.10

118 schöne Ferien

Bild 11.25
Typischer ECR 810-Rechnungscoupon

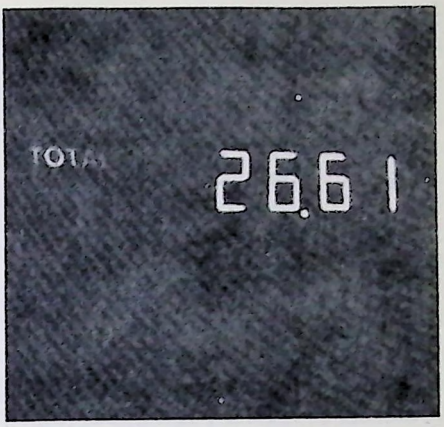


Bild 11.26
ECR 810-Anzeige

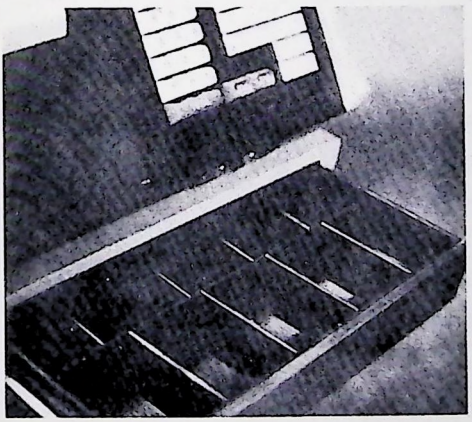


Bild 11.27
ECR 810-Kassenlade

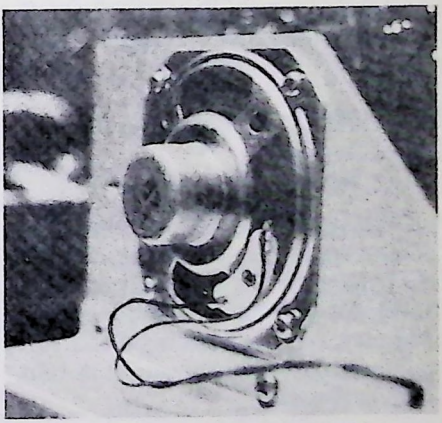


Bild 11.28
ECR 810-Alarmanlage

ECR 810-Schlüsselschalter
(Bild 11.29)

Der Schlüsselschalter stellt eine Absicherung gegen unerlaubte Manipulationen dar, und man kann damit auch die Betriebsart des Terminals wählen. Die in Bild 11.29 dargestellten fünf Betriebsarten des Terminals sind AUS (OFF), Management-(MGMT), Kassieren-(CASHIER), Nacht-(NIGHT) und Wartungsbetrieb (MAINT).

Durch Abfragen des Zustandswortes ist der Prozessor in der Lage, festzustellen, in welcher Betriebsart sich das Terminal befindet. In der Betriebsart "Management" hat der Verkaufsstellenleiter über das Tastenfeld Zugriff zu Berichten und anderen Managementfunktionen. In der Betriebsart "Nacht" werden die Register gesperrt, ausgenommen für Übertragungen zum Zentralrechner. Die normale Betriebsart "Kassieren" ist für reguläre Verkaufsfunktionen vorgesehen.

ECR 810-Überwachungs- und Index-Anzeige
(Bild 11.30)

Am Tastenfeld befindet sich ebenfalls die Überwachungs- und Index-Anzeige. Hier handelt es sich um ein abgesetztes Paneel, das die Bedienperson informiert und leitet. Eine Aufstellung dieser Informationen und Anweisungen ist in Bild 11.30 dargestellt.

Zusammenfassung

Dieses Kapitel behandelte die Funktionen des Kassen-Terminals und die spezifische Entwicklung eines Terminals. In einem solchen Anwendungsfall liefern Mikroprozessoren die Rechnerleistung, die früher von einem Minicomputer erfüllt wurde. Unter Verwendung eines POS-Terminals können diese Funktionen in wirtschaftlicherer Form und mit größerer Zuverlässigkeit durch einen Mikroprozessor übernommen werden.

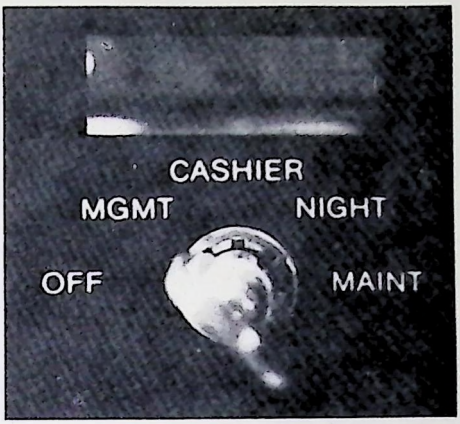


Bild 11.29
ECR 810-Schlüsselschalter

- Gesamtsumme
- Wechselgeld
- Zu wenig Papier
- Rechnungsbetrag
- Fehler
- Spezial
- Management-Information

Bild 11.30
ECR 810-Überwachungs- und Index-Anzeige

KAPITEL 12

MIKROPROZESSOR-ANWENDUNGEN ANDERE TERMINAL-ARTEN

Dieses Kapitel beschreibt die Anwendung von Mikroprozessoren in verschiedenen Terminals und insbesondere den Einsatz in einem Mehrzweck-Datensichtgerät.

Mikroprozessoranwendungen: Andere Terminal-Arten

In diesem Kapitel wird ein komplexes Mehrzweck-Datensichtgerät untersucht, und zwar innerhalb eines Netzes, in dem Terminals einen hohen Grad an "lokaler Intelligenz" besitzen müssen, um einer großen Zahl von Benutzeranforderungen gerecht zu werden.

Diskutiert werden drei wichtige Aspekte eines komplexen Mehrzweck-Datensichtgerätes: die funktionellen Spezifikationen dieser Einheit, der Aufbau, der zur Erfüllung dieser Forderungen realisiert wurde, einschließlich der logischen Ausführung, sowie einige der verschiedenen Erweiterungen des Terminals, die sich durch den Einsatz eines Mikroprozessors ergeben haben.

Grundsätzliche Anforderungen an den Entwurf eines Datensichtgerätes
(Bild 12.1)

Die grundsätzlichen Anforderungen an ein Mehrzweck-Datensichtgerät sind in Bild 12.1 zusammengefaßt.

Verwendetes Verfahren für die grafische Anzeige von alphanumerischen Zeichen und Symbolen
(Bild 12.2)

Ein Verfahren zur grafischen Anzeige von alphanumerischen Zeichen in einem Raster von 5 x 6 Punkten in einer 7 x 9-Punktmatrix ist in Bild 12.2 dargestellt. Damit können Figuren, wie Strichsymbole oder andere Bilder, angezeigt werden.

Definition und Einsatz von Zusatz-Bytes
(Bild 12.3)

Der Anzeigeschirm ist in mehrere Bildflächen oder "Felder" zu unterteilen, die eine Trennung der verschiedenen Informationsklassen ermöglichen. Diese Trennung wird im Terminal durch Verwendung einer Technik realisiert, die "Zusatz-Bytes" einsetzt. Bei diesen Zusatz-Bytes handelt es sich um Zeichen, die zwei Dinge bewirken: Sie belegen den Speicher und definieren die Merkmale der folgenden Zeichen, bis ein weiteres Zusatz-Byte aufgenommen wird. Ein Zusatz-Byte kann ein Feld auf eine der folgenden Arten definieren: hohe Leuchtstärke, geringe Leuchtstärke, blinken oder ausblenden. Ein ausgeblendetes Feld wird oft für Sicherheitszwecke benutzt.

Ein Feld kann entweder grafisch oder alphanumerisch genutzt werden. Darüber hinaus kann die Eingabe in ein Feld, oder Felder, über die Tastatur frei oder gesperrt sein, übertragbar oder nichtübertragbar zu einem Zentralcomputer und alphanumerisch oder nur numerisch aufgebaut sein. Die ersten beiden Merkmale beeinflussen die Anzeige des Speicherinhalts, während die letzten drei Schriftzeichen das Zusammenwirken der Übertragungsleitung und des Tastenfeldes mit dem Anzeigespeicher steuern.

Das Datensichtgerat (VDU) muß mit den Video-Terminals der IBM 3270-Serie kompatibel sein. Die Anforderungen an ein Datensichtgerat umfassen:

Format der Anzeigefläche:
- Mindestschirmgröße 14"
- Format von 24 Zeilen und 80 Schriftzeichen pro Zeile
- Insgesamt 1920 Schriftzeichen auf der Schirmfläche
- Schriftzeichen gebildet durch 7 x 9-Punktmatrix
- Fähigkeit zur Anzeige von USASCII-Groß- und Kleinschriftzeichen
- Möglichkeit der Darstellung von grafischen Zeichen/Symbolen

Tastenfunktionen

Datenubertragungsprotokoll

Bild 12.1
Grundsätzliche Anforderungen beim Entwurf eines Datensichtgerätes

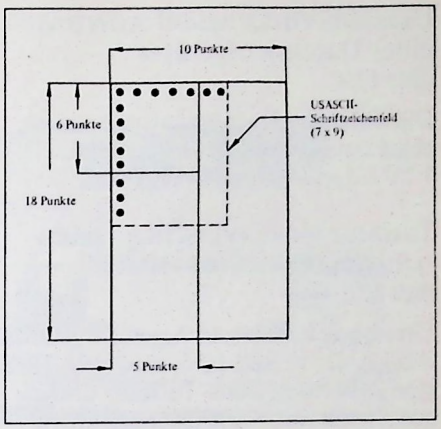


Bild 12.2
Verfahren zur Anzeige von alphanumerischen Zeichen und grafischen Symbolen

Zusatz-Bytes

Belegung des Speichers

Festlegung der Merkmale sämtlicher nachfolgender Zeichen

Bild 12.3
Definition und Einsatz von Zusatz-Bytes

Datenübertragungsanforderungen eines Datensichtgerätes
(Bild 12.4)

Die Datenübertragungsanforderungen eines Datensichtgerätes sind in Bild 12.4 aufgelistet.

Tastatur eines typischen Datensichtgerätes
(Bild 12.5)

Eine typische Datensichtgerät-Tastatur, ähnlich der in Bild 12.5, muß 108 Tasten aufweisen. Diese Tastatur umfaßt den Standard-USASCII-Zeichenvorrat, einer aus 11 Tasten bestehenden Gruppe für numerische Auffüllung, eine Löschtaste für Buchstaben, eine Einfügungstaste, vier Annuliertasten, fünf CURSOR-Positionstasten, vier Programmhinweistasten und fünfzehn Programmfunktionstasten.

Statusanzeigen des Datensichtgerätes
(Bild 12.6)

Wie in Bild 12.6 dargestellt, besitzt das Datensichtgerät eine Gruppe von Statusanzeigen zur Anzeige der Betriebsbedingungen des Terminals. Diese Statusanzeigen umfassen Netz-EIN (Power ON), Verfügbarkeit des Systems (SYSTEM AVAILABLE), Terminal angewählt (TERMINAL SELECTED), Tastatureingabe gesperrt (INPUT DISABLED) und Datenübertragungsfehler (DATA ERROR). Darüber hinaus sind vier Zwischenstatusanzeigen sowie eine akustische Alarmeinrichtung vorhanden, die über die Software als Zentralsystem aktiviert werden. Hierbei handelt es sich um programmierfähige Zusätze für den Benutzer, der diese Funktionen wahlweise, entsprechend seinen Erfordernissen bestimmen kann. Bei einem in dieser Form aufgebauten System hat der Benutzer die Möglichkeit, diese zusätzlichen Anzeigen über die Software (Programmierung) zu erreichen.

Eine weitere wichtige Entwurfsanforderung ist die Erweiterungsfähigkeit. Hierunter wird verstanden, daß Vorkehrungen innerhalb dieses Terminals getroffen sind, daß es die Steuerung von verschiedenen Peripheriegeräten, wie zum Beispiel einem synchronen Modem oder einem Drucker übernehmen kann.

Wie sämtliche dieser Forderungen vermuten lassen, ist eine sehr komplexe und universelle Einheit erforderlich. Nach Kenntnis dieser Systemanforderungen können die Architektur des Terminals sowie die Gründe für verschiedene Entwurfsentscheidungen bei der Festlegung dieser Architektur weiter untersucht werden.

Vereinfachtes Datensichtgerät
(Bild 12.7)

Bild 12.7 zeigt ein Gesamtsystem, das die meisten Anforderungen erfüllen kann. Eine detailliertere Version dieses Systems wird zum Abschluß dieses Kapitels vorgestellt (Bild 12.A). Die funktionellen Erfordernisse eines intelligenten Terminals lassen sich in zwei bestimmte Kategorien unterteilen. Kategorie 1 umfaßt die Funktionen, die zur Anzeige der Speicherinhalte gehören. Diese Funktionen werden ohne Bezug auf die Anwendung des Terminals festgelegt. Auch besitzen diese Wiederholcharakter und treten mit einer sehr hohen Datenrate auf.

Die zweite Gruppe von Funktionen umfaßt jene, die die Datenübertragungs-Schnittstelle, das Tastenfeld, die Statusanzeigen und die Erweiterungsschnittstelle umfassen. Diese Funktionen laufen bei einer verhältnismäßig geringen Geschwindigkeit ab, sie sind jedoch in Echtzeit zu verarbeiten und hängen alle in großem Maße von der Anwendung des Terminals ab. Aufgrund dieser Erfordernisse ist die Logik des Terminals unterteilt. Hier handelt es sich um ein festverdrahtetes Steuergerät zur Erzeu-

- Abfragbares Terminal (adressierfähig)
- Steuerung über asynchrones 1200-Baud-Modem oder 1200-, 2400-, 4800- oder 9600-Baud-"Ortsleitung"

Bild 12.4
Datenübertragungsanforderungen eines Datensichtgerätes

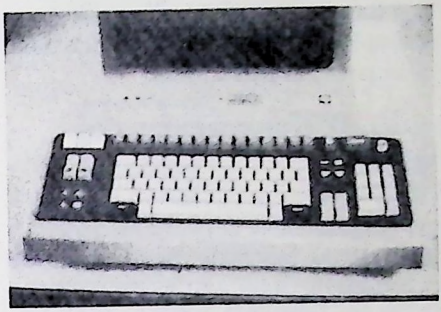


Bild 12.5
Typische Tastatur eines Datensichtgerätes

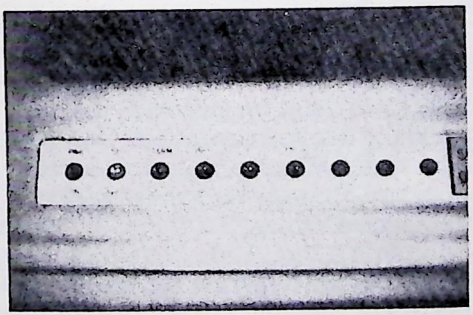


Bild 12.6
Statusanzeigen des Datensichtgerätes

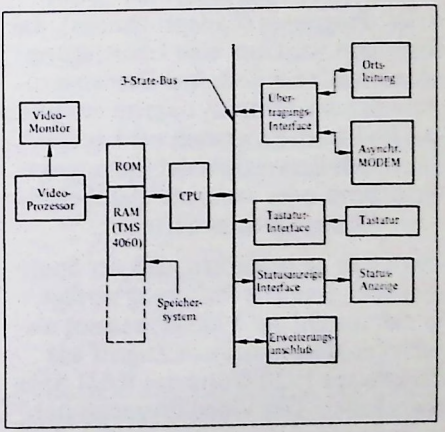


Bild 12.7
Vereinfachtes Datensichtgerät

gung des Bildschirminhalts, dem auf der anderen Seite ein Mikroprozessor und die dazugehörige Software zur Verfügung stehen, um die verschiedenen Terminal-Eingabe/Ausgabeoperationen zu steuern. Beide Einheiten teilen sich das gemeinsame Speichersystem.

Betriebssystemprogramm des Terminals
(Bild 12.8)

Eine Untersuchung der verschiedenen Untersysteme innerhalb dieses Abschnittes beginnt mit dem Betriebssystemprogramm des Terminals, welches in einem RAM gespeichert ist, wie in Bild 12.8 dargestellt. Im Grundsystem sind 8 K Worte enthalten. Hierbei ist ein nichtflüchtiger Speicher (ROM) für die Initialisierung bzw. die Einschaltroutine vorzusehen. Um dies zu erreichen, enthält ein kleiner Teil des ROMs ein "bootstrap"-Programm (Ladeprogramm), das ausgeführt wird, um eine Übertragung (down-load) des Terminal-Betriebsprogramms vom Zentralsystem zu erwirken. Das Bootstrap-Programm wird ausgeführt nach Einschalten der Spannungsversorgung, oder wenn ein nichtbehebbarer Terminal-Fehler auftritt.

Hierbei ist zu bemerken, daß der Speicher über doppelte Pufferung verfügt, so daß sowohl der Video-Prozessor als auch der Mikroprozessor Zugriff auf die unteren 1920 Worte des RAM-Speichers haben. Der Video-Prozessor hat zu diesem Teil des RAM-Speichers Zutritt, um auf dem Bildschirm ständig den neuesten Stand wiederzugeben. Der Mikroprozessor hat zu diesem Teil Zugriff, um die anzuzeigenden Informationen verändern zu können. Der restliche Inhalt des RAM-Speichers ist dann für Programmspeicherung verfügbar. Der RAM-Speicher ist realisiert mit dem dynamischen NMOS-RAM TMS 4060.

Daten- und Adressleitungen des Systemspeichers
(Bild 12.9)

Bild 12.9 zeigt die Adresse (A-Bus), die sowohl vom Mikroprozessor als auch vom Video-Prozessor kommt und die die Speicher-Adressleitungen sowie die Speicherblocklogik treibt. Der Eingabe- und Ausgabedatenbus (D-Bus IN/OUT) ist gepuffert, um die jeweiligen Ausgangslasten zu treiben und 3-State-Ausgänge (D-Bus OUT) zu erhalten. Obwohl dynamische RAMs verwendet werden, enthält der Speicher keine Auffrischlogik, da dies der Video-Prozessor automatisch gewährleistet.

Funktionsblöcke des Video-Prozessors
(Bild 12.10)

Die einzelnen Funktionsblöcke des Video-Prozessors sind in Bild 12.10 gezeigt. Diese schließen ein die Zeitbasis, die Speichereingabesteuerung, den Datenpuffer und die Ausgabesteuerung.

Entwicklung des Anzeigerasters
(Bild 12.11)

Bild 12.11 zeigt im einzelnen die Entwicklung des Anzeigerasters. Das Raster wird verschachtelt abgetastet, da ein Standard-Monitor verwendet wird und 24 Zeilen à achtzig 7 x 9 Punkte-Zeichen erforderlich sind. Unter verschachtelter Abtastung wird hierbei verstanden, daß bei jedem Halbbild nur jede zweite Punkt-Zeile abgebildet wird.

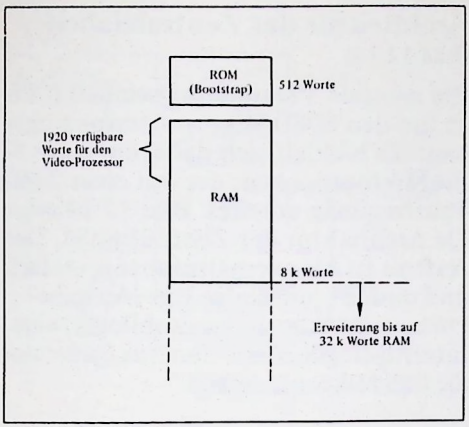


Bild 12.8
Organisation des Betriebssystemprogramms

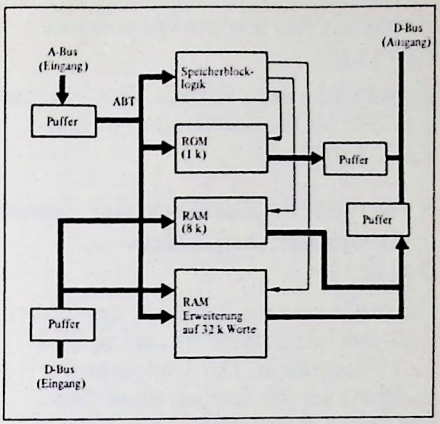


Bild 12.9
Daten- und Adressleitungen des Systemspeichers

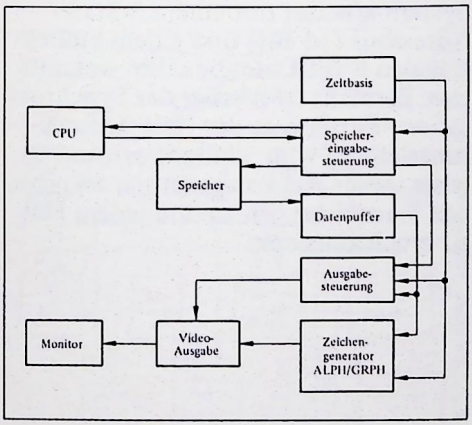


Bild 12.10
Funktionsblöcke des Video-Prozessors

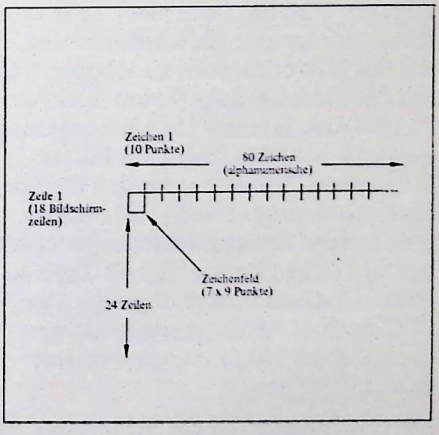


Bild 12.11
Entwicklung des Anzeigerasters

Zeitbasis des Video-Prozessors
(Bild 12.12)

Einzelheiten des Zeitbasis-Video-Prozessors sind im Blockdiagramm in Bild 12.12 dargestellt.

Verschiedene Zustände der Steuerlogik des Umlaufpuffers
(Bild 12.13)

Die verschiedenen Zustände der Steuerlogik des Umlaufpuffers sind in Bild 12.13 dargestellt. Der Umlaufpuffer speichert die 80 Zeichen einer Zeile. Der Inhalt dieses Puffers zirkuliert 9 mal, um das Bitmuster für die 9 Rasterabtastzeilen einer Zeichenreihe zu entwickeln. Die Steuerlogik wird dazu verwendet, das Laden des Puffers aus dem RAM-Speicher zu steuern, und zwar am Ende einer jeden Zeichenreihe von 80 Zeichen. Wird das Ende einer Zeile erkannt, erfolgt eine Halt-Anforderung, um den Mikroprozessor zu stoppen und um den direkten Zugriff zum Speicher zu erwirken. Ist diese Halt-Anforderung quittiert, wird das Laden des Puffers freigegeben. Außerdem werden die Speicheranforderung (Lesen) und die entsprechenden Speicheradressen generiert, um 80 Zeichen für die nächste Zeile zu laden. Ist dieser Prozeß abgeschlossen, wird die Halt-Anforderung zurückgesetzt und der Mikroprozessor startet erneut.

Die Geschwindigkeit der Steuerlogik ist so bemessen, daß sich die Mikroprozessor-Betriebszeit nur um ca. 10% verringert. Am Ende dieses Kapitels ist eine Zeichnung zu finden (Bild 12.B), die die Verbindungen dieser Komponenten zueinander zeigt. Der Video-Prozessor sowie das Speichersystem enthalten eine große Anzahl von Logikbausteinen. Tatsächlich sind etwa 125 der 210 im gesamten Terminal verwendeten Schaltkreise für den Video-Prozessor erforderlich.

Architektur der Zentraleinheit
(Bild 12.14)

Die zentrale Verarbeitungseinheit (CPU) ist um den 8080-Mikroprozessor aufgebaut. Es handelt sich dabei um einen 8-Bit-Mikroprozessor, der mit einer 2-MHz-Taktfrequenz arbeitet. Bild 12.14 zeigt die Architektur der Zentraleinheit. Der Aufbau ist hier verhältnismäßig einfach und umfaßt nur einige Ein-/Ausgabe-Ports, eine Adressenauswahllogik, eine Interruptlogik sowie den Taktgenerator für den Mikroprozessor.

Mikroprozessorsystem und Pufferung
(Bild 12.15)

Das Verarbeitungssystem, der Mikroprozessor, sowie die Pufferung sind als Diagramm in Bild 12.15 dargestellt. Das System arbeitet mit einem 3-State-Adressbus (16 Bit) und einem bidirektionalen 8-Bit-Datenbus. Ein wesentliches Element ist hierbei das Synchronisations-Flipflop in der Halt-Anforderungsleitung vom Video-Prozessor. Signale dieser Art von externen asynchronen Vorrichtungen sind in jedem Fall zu synchronisieren.

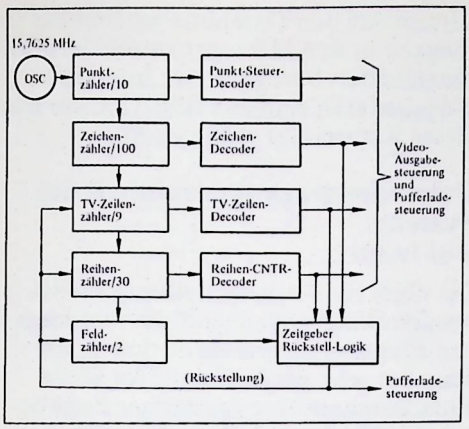


Bild 12.12
Zeitbasis des Video-Prozessors

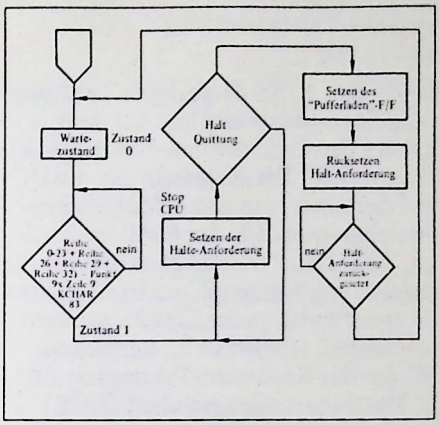


Bild 12.13
Verschiedene Zustände der Steuerlogik des Umlaufpuffers

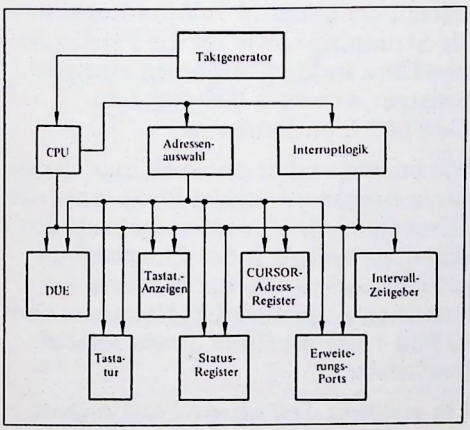


Bild 12.14
Architektur der Zentraleinheit

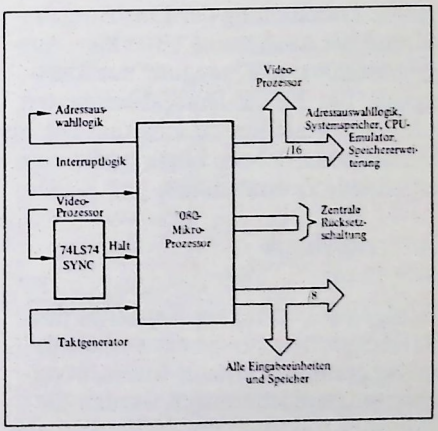


Bild 12.15
Mikroprozessorsystem und Pufferung

System-Taktgenerator
(Bild 12.16)

Der in Bild 12.16 dargestellte Taktgenerator des Systems arbeitet mit dem 'LS163-Baustein, der den 12-MHz-Takt herunterteilt. Die Ausgänge des 'LS163 sind decodiert, um den erforderlichen Zweiphasentakt für den 8080 zu erhalten. TTL-Takte werden ebenfalls für die gesamte Logik erzeugt, um die Synchronisation für das ganze System aufrecht zu erhalten. Hierbei ist zu bemerken, daß die vier Baudraten-Taktsignale für die Datenübertragungseinheit (DÜE) von einem 4-MHz-Takt heruntergeteilt werden.

Adress-Auswahllogik
(Bild 12.17)

Die Adress-Auswahllogik, die in Bild 12.17 dargestellt ist, erfüllt zwei Aufgaben: Decodierung der Ein-/Ausgabeadresse bei Ausführung eines Ein-/Ausgabebefehls und Erzeugung der Taktsignale, um die Ein-/Ausgabeeinheiten auf den 3-State-Bus zu schalten. Die bis hierher beschriebene Logik bildet eine arbeitende Zentraleinheit.

Interruptlogik
(Bild 12.18)

Infolge der begrenzten Kapazität des ROM-Speichers, sowie der verhältnismäßig geringen Echtzeit-Interruptverarbeitungsanforderungen werden die Interrupt-Vektoren des 8080 nicht eingesetzt. Für alle Interrupts wird nur ein Interrupt-Vektor benutzt, und die Prioritäts-Decodierung erfolgt in der Hardware der Zentraleinheit. Bild 12.18 zeigt die Interruptlogik. Der Empfang irgendeines Interrupts führt zu einem Interruptsignal (INT) zum Mikroprozessor.

Läßt der Mikroprozessor einen Interrupt zu, so wird der Interrupt-Adreßvektor geladen und die Software führt einen Eingabebefehl aus. Dies führt zu einem Taktsignal, das die Interruptadresse auf den Datenbus schaltet zur Eingabe in den Mikroprozessor. Weitere Einzelheiten bezüglich der Interrupt-Eingabedaten sind aus Bild 12.C am Ende dieses Kapitels zu entnehmen.

Datenübertragssteuereinheit des Systems
(Bild 12.19)

Die noch für die Beschreibung verbleibenden Baugruppen sind die verschiedenen Eingabe/Ausgabefunktionen, die einander sehr ähnlich sind. An Stelle einer Beschreibung sämtlicher Funktionen wird hier nur auf die Datenübertragungssteuereinheit eingegangen, der sowohl Eingabe als auch Ausgabefunktionen obliegen.

Die als Blockschaltung in Bild 12.19 dargestellte Datenübertragungssteuereinheit ist im Aufbau sehr einfach, da nahezu der gesamte Teil der Logik für die Steuerung sowie für die Parallel-Serien-Umwandlung auf einen einzigen Baustein, wie zum Beispiel dem TMS 6011, enthalten ist.

Wie bereits vorher erwähnt, sind die anderen Eingabe/Ausgabe-Ports innerhalb dieses Systems ähnlich aufgebaut. Einzelheiten bezüglich der Eingabe/Ausgabeadressen sowie der Struktur der einzelnen Schnittstellen-Datenbits sind in Bild 12.D am Ende dieses Kapitels beschrieben.

Ein weiterer Teil dieses Terminals ist die Software, die in einer modularen Form entwickelt wurde. Wenn auch dieser Aufbau nicht das Optimum der Programmlänge erreicht, so liefert es doch ein universelles und leicht zu modifizierendes System. Nach Erstellung der Definition sowie des Ablaufdiagrammes für jedes Modul erfolgt die Codierung in Assembler-Sprache.

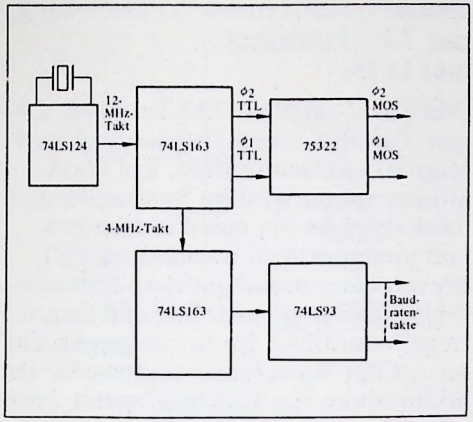


Bild 12.16
Systemtaktgenerator

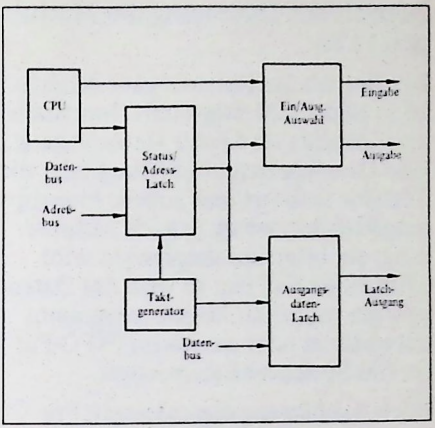


Bild 12.17
Adreß-Auswahllogik

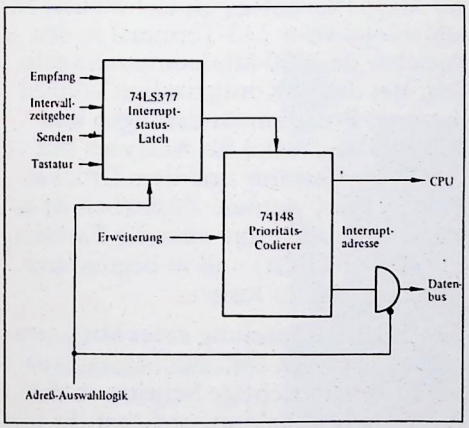


Bild 12.18
Interruptlogik

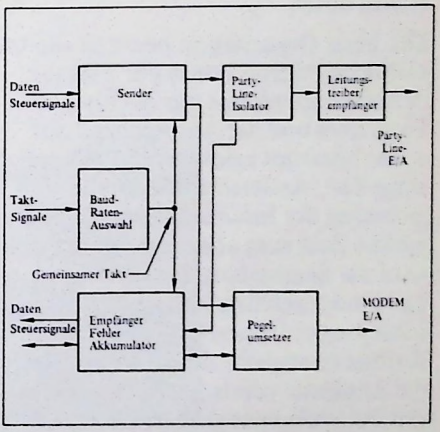


Bild 12.19
Datenübertragungssteuereinheit

Betrieb des Dateneingabe-Moduls
(Bild 12.20)

Der Betrieb des Dateneingabe-Moduls ist in Bild 12.20 dargestellt. Innerhalb eines Moduls wird jeder Dateneingang, jede Datenübertragungsleitung oder die Tastatur aktiviert und andere Interrupts ausgeblendet, wenn, je nach Anforderung, ein Interrupt empfangen wird. Tritt dieser Fall ein, so wird das Datenzeichen vom Ein-/Ausgabeprogramm entweder in oder aus einem FIFO-Puffer (im Speicher) transportiert.

Nach Ausführung dieses kurzen Programms wird der Interrupt-Eingang für neue Anforderungen freigegeben und das erforderliche Interruptprogramm gestartet. Der Mikroprozessor übernimmt anschließend die Daten vom FIFO und führt die geforderten Operationen durch.

Bei dieser Organisation benötigt die tatsächliche Interpretation der Eingabe der Interruptadresse nur ein kleines Programm und die Ansprechzeit auf einen Interrupt erfordert nur eine geringe Zeit. Andererseits kann die Verarbeitung der Interruptdaten einen großen Zeitraum erfordern. Demzufolge wird die Bearbeitung der Interruptdaten dann durchgeführt, wenn Interruptanforderungen freigegeben sind. Dieser Aufbau ermöglicht der Zentraleinheit die Annahme von Interruptanforderungen mit einer weit größeren Rate im Vergleich zu der sonst möglichen. Nach Definition und Codierung sämtlicher Moduln wird ein Cross-Assembler benutzt, um die vollständigen Programme zu erhalten.

Software-Entwicklungssystem
(Bild 12.21)

Um eine Software-Entwicklung vor der Verfügbarkeit der Hardware zu ermöglichen, wird ein Software-Entwicklungssystem aufgebaut, das aus den in Bild 12.21 dargestellten Anlageteilen besteht.

Grundsystem unter Verwendung des 733-Terminal
(Bild 12.22)

Bild 12.22 zeigt ein 733-Terminal, das eine Tastatur, einen Drucker und zwei Magnetkassetten umfaßt. Ein 960A-Minicomputer wird im System benutzt, da es sich hier um einen vielseitigen und preisgünstigen Rechner handelt, der mit einer umfangreichen Software-Bibliothek ausgestattet ist und demzufolge wesentliche Einsparungen bei dieser Art der Anwendung sicherstellt. Das Endprodukt, das Datensichtgerät, kann mit einem anderen Terminal-Typ nachgebildet werden, oder es kann sich hierbei auch um einen Prototyp oder um ein Entwicklungsmuster handeln.

Während des Betriebes wird das assemblierte Programm innerhalb des Systems auf Magnetkassette gespeichert und anschließend vom 733-Terminal in den Speicher des 960-Minicomputers geladen. Bei dieser Konfiguration können begrenzt Programmänderungen und Speicherausdrucke für Analysen mit der 733er Tastatur und dem Drucker durchgeführt werden. Zusätzlich ist es möglich, Programme über die Tastatur einzugeben (Edit) und in begrenztem Umfang auch zu testen.

Bei dieser Anwendung eines Mikroprozessors in einem Datensichtgerät wurden mehrere wichtige Schritte in der Software-Entwicklung sichtbar. Es wurde hier die Software in Moduln definiert, codiert in Assembler-Sprache, mit einem Cross-Assembler assembliert und korrigiert mit einem Software-Entwicklungssystem. Der wesentliche Vorteil dieses Software-Entwicklungssystems liegt darin, daß es aus direkt lieferbaren Baugruppen zusammengestellt werden kann, bevor das endgültig komplette System verfügbar ist.

Eine Ausführung dieses Terminal-Typs mit einem Mikroprozessor, verglichen mit einer festverdrahteten Steuerung,

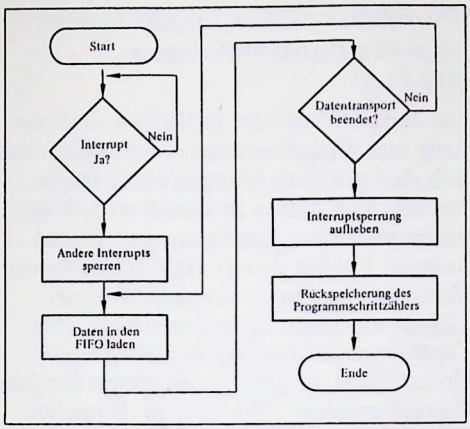


Bild 12.20
Betrieb der Dateneingabe-Moduln

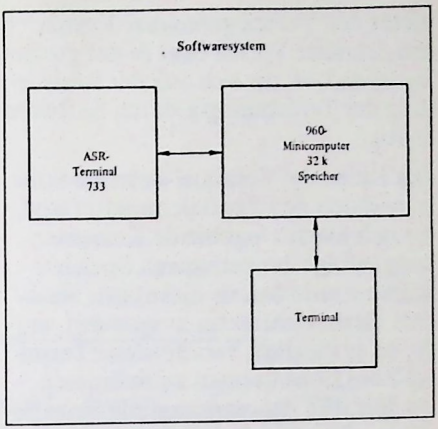


Bild 12.21
Software-Entwicklungssystem

Bild 12.22
Grundsystem unter Verwendung der Terminals 733

bietet den Vorteil geringerer Kosten. Ein primärer Vorteil liegt in der großen Vielseitigkeit, die sich aus der Realisierung der Terminallogik durch Software ergibt.

Das Laden der Terminal-Betriebs-Software durch den Zentralcomputer und die sich hieraus ergebende Kostensenkung infolge der geringeren Speicherkosten wurde bereits diskutiert. Nachdem diese Architektur ausgewählt wurde, ist es möglich, verschiedene Terminal-Funktionen erneut zu definieren, um spezielle Anwendungsfälle einzubeziehen. So wären beispielsweise Tastenfunktionen erneut zu definieren, um ein neues Merkmal wie zum Beispiel die Einführung von Datenseiten bei großen Datenmengen mit einer Seitenfolgetaste. Auf diese Weise kann durch einen einfachen Wechsel der Tastenbeschriftung sowie der Ergänzung der Software dieser äußerst leistungsfähige Zusatz einbezogen werden.

In die Software einbezogen können ebenfalls lokale Masken. In vielen Anwendungsfällen sind Situationen vorhanden, in denen eine "Maske" oder Schirmformat aufgerufen werden muß, und bei denen die Eingabefelder mit dem Namen des Feldes bezeichnet werden, wobei vom Benutzer erwartet wird, die Leerstellen auszufüllen. Bei dieser Betriebsart können nahezu 50% des Datenverkehrs zum Zentralcomputer das Abrufen von Masken beinhalten. Werden die am häufigsten benutzten Masken im Speicher des Terminals gespeichert, so können diese ohne Zuhilfenahme des Zentralsystems aufgerufen werden, so daß die effektive übertragene Datenmenge zwischen Sichtgerät und Zentralcomputer nahezu verdoppelt wird. Hierdurch kann eine größere Anzahl von Terminals mit dem gleichen Zentralsystem verbunden werden, und zwar bei einer Verringerung der Gesamtsystemkosten.

Grundsystem für lokale Datenerfassungsanwendungen (Bild 12.23)

Die Möglichkeit der örtlichen Aufbereitung von Eingabedaten stellt wahrscheinlich den größten Nutzen einer Implementierung dieses Systems mittels der Mikroprozessor-Software dar. Somit können Felder durch viele Algorithmen überprüft werden, wie zum Beispiel Länge, alphanumerischer Inhalt, Prüf-Digit-Berechnung sowie viele andere Tests, die sich auf die einzelnen Eingabedaten beziehen. Werden die Eingaben örtlich, d.h. vor der Übertragung überprüft, dann wird die Übertragungszeit zum Zentralcomputer verringert, da hier keine Fehlermeldungen auftreten. Selbstverständlich arbeitet man bei den Hauptanwendungen der lokalen Datenerfassung mit einer Stapelverarbeitung, bei der der Benutzer Informationen in den lokalen Speicher für eine spätere Übertragung einer großen Anzahl von Daten zum Zentralcomputer eingibt.

Betrachten wir nunmehr die in Bild 12.23 gezeigte Konfiguration. Das Erweiterungs-Port des Terminals ermöglicht den Anschluß von einer Vielzahl von Geräten. Innerhalb dieser Konfiguration ist ein mit zwei Geschwindigkeiten arbeitendes asynchrones Modem mit einer Wählleitung verbunden und verfügt über eine automatische Wähl- und Antwortsteuerung. Das System enthält außerdem eine örtliche Speichereinheit.

Diese Konfiguration eignet sich für die Erfassung von Katalogbestellungen, bei der die Bestelldaten täglich erfaßt und aufbereitet werden. Nachts wählt das Terminal automatisch den Zentralcomputer an und überträgt die erfaßten Daten zur weiteren Verarbeitung. Wird diesem System ein lokaler Drucker zugeordnet, so kann der Zentralcomputer Daten in den örtlichen Speicher übertragen, die nach erfolgter Abschaltung

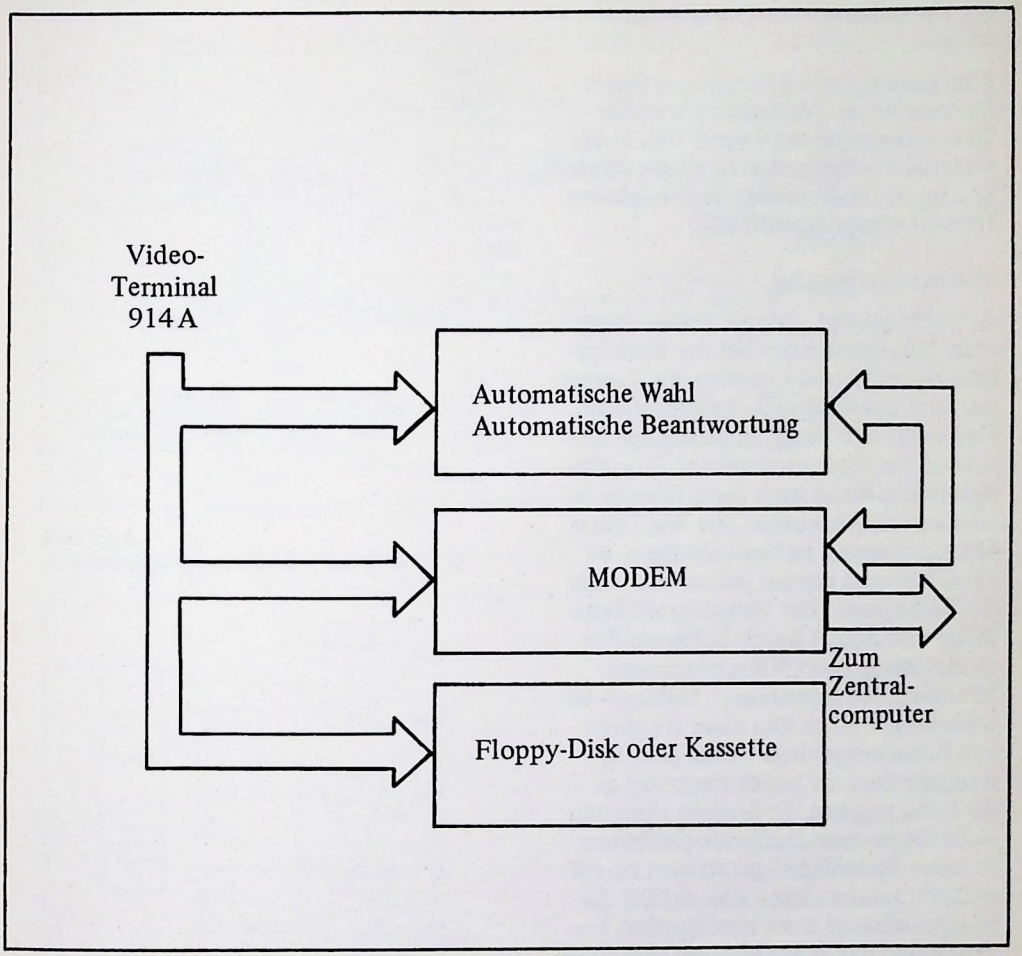


Bild 12.23
Grundsystem für örtliche Datenerfassungs-anwendungen

der Übertragungsverbindung ausgedruckt werden.

Eine naheliegende Erweiterung dieses Systems ist die Zuschaltung weiterer Datensichtgeräte des Typs 914A in der Minimal-Konfiguration zu einem Zentral-Sichtgerät (Master) über dessen lokalen Datenübertragungsanschluß.

Zusammenfassung

Es wurde gezeigt, daß die Anwendung eines Mikroprozessors bei der Entwicklung eines Datensichtgerätes die Kompliziertheit des Entwurfs verringern kann. Dies wirkt sich direkt in geringeren Kosten und in einer verbesserten Zuverlässigkeit aus. Wenn auch diese Gründe bereits ausreichen würden, die Wahl eines Mikroprozessors zu beschleunigen, so handelt es sich hierbei jedoch nicht um die Wichtigsten. Der Hauptvorteil liegt in der Flexibilität durch Software-Programmierung eines Mikroprozessors. Mit einem Mikroprozessor, ladbaren Betriebsprogrammen und einer Auswahl von Peripheriegeräten wurde dem Systemingenieur ein neues Werkzeug in die Hand gegeben. Er braucht nunmehr nicht länger neue Hardware-Einheiten für neue Systemkonfigurationen zu entwickeln, sondern kann sich auf die Zusammenstellung eines intelligenten Terminals und das Schreiben von verhältnismäßig kosteneffektiven neuen Programmteilen beschränken. Somit erhält dieser äußerst leistungsfähige und kosteneffektive Werkzeuge zur Entwicklung neuer Systemanwendungen.

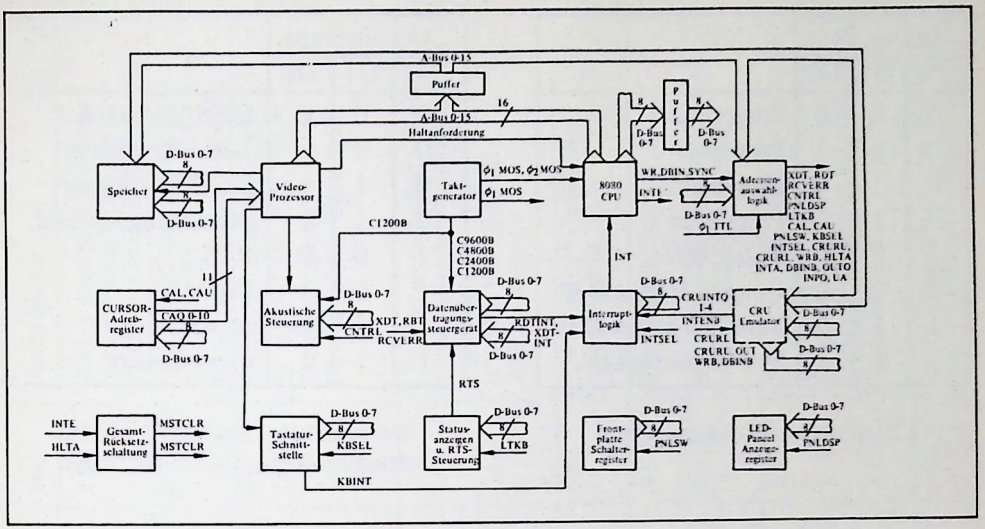


Bild 12.A
Blockdiagramm des 914 A-Steuergerätes

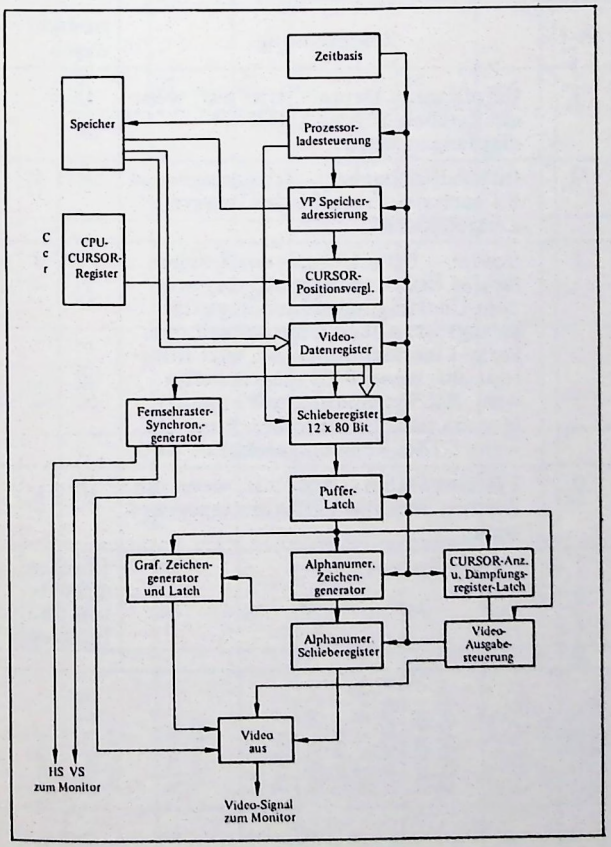


Bild 12.B
Blockdiagramm der Video-Prozessor-Steuerung und Dateneinheit

Akkumulator			
D2	D1	D0	
0	0	0	Programm 4
0	0	1	Programm 3
0	1	0	Programm 2
0	1	1	Programm 1
1	0	0	Datenfehler
1	0	0	Eingabe freigegeben
1	1	0	Terminal angewählt
1	1	1	System verfügbar

LED Paneel Anzeigeadressen

Akkumulator			
D2	D1	D0	
0	0	0	DXS-Betriebsart
0	0	1	2260-Betriebsart
0	1	0	3270-Betriebsart
0	1	1	Schriftzeicheneinf.
1	0	0	RTS
1	0	1	ENAVD
1	1	0	nicht belegt
1	1	1	nicht belegt

beleuchtete Tasten, RTS und freigegebene Video-Adressen

Priorität	Signal	IA 3	IA 2	IA 1	Beschreibung	gelöscht durch
1	RDTINT	1	1	1	Empfangene Daten – tritt auf, wenn ein Zeichen durch COMM.I/F empfangen wurde.	IN 0
2	ITVINT	1	1	0	Intervall-Zeitgeber – tritt annähernd 8 s nach der Freigabe des Intervall-Zeitgebers auf.	OUT 4
3	RDTINT	1	0	1	Sender – öffnet, wenn ein Zeichen für das Übertragungspufferregister zum Übertragungsschieberegister gelangt ist; sofern letzteres mit der Party-Line verbunden ist, tritt Interrupt auf, wenn RTS eingeschaltet wird. Bei Verbindung mit einem Modem tritt die Unterbrechung ein wenn CTSE-Zeile eingeschaltet wird.	OUT 0
4	RBINT	1	0	0	Tastatur-Daten – tritt auf, wenn ein Zeichen über die Tastatur eingegeben wird.	IN 3
5 6	CRUINT 1 CRUINT 2	0 0 0 0	1 1 0 0	1 0 1 0	CRU-Interrupt	Löschen des entspr. CRU-Interrupts

Bild 12.C
Interrupt-Prioritäten und Adressen

Bild 12-B
Eingabe/Ausgabe-Adressen

Befehl	Modul	Akkumulator-Inhalt								Beschreibung
		D7	D6	D5	D4	D3	D2	D1	D0	
OUT 0	Comm.I/F Sender	IXBL	XDT 6	XDT 5	XDT 4	XDT 3	XDT 2	XDT 1	XDT 0	IXBL = sperrt ladendes Übertragungspuffer (XBR). Wenn IXBL = 0, (XDTONXDTE)→XBR
OUT 1	Akkust. Anzeige	X	X	X	X	X	X	KOLK	ARLRM	KOLK = 125 ms, 1200-Hz-Ton, ARLRM = 500 ms, 2400-Hz-Ton, autom. Rückst. durch Hardware
OUT 2	LED-Paneel	X	X	X	B	X	D2	D1	D0	Setzt LED, adressiert durch D0 -D2 auf den Wert von B
OUT 3	beleucht. Tasten, RTS + Freischaltung	X	X	X	B	X	D2	D1	D0	Setzt Zeile adressiert durch D0-D2 auf den Wert von B
OUT 4	Intervall-Zeitgeber	X	X	X	X	X	X	X	ITE	ITE = Intervall-Zeitgeber freigegeben. Zeitgeberzurücksetzung unabhängig von Wert ITE
OUT 6	8 LSB'S CURSOR-Adresse	CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0	(CA0-CA7) = CURSOR-Adressregister
OUT 7	3 LSB'S CURSOR-Adresse	X	X	X	X	X	CA10	CA9	CA8	(CA8-CA10) = CURSOR-Adressregister
IN 0	Comm.I/F Empfänger	0	RDT6	RDT5	RDT4	RDT3	RDT2	RDT1	RDT0	Empfangene Daten
IN 1	Empfänger Fehlerakkumulator	1	1	1	1	1	RPE	RPE	RTE	RPE = Formatfehler, RPE = Paritätsfehler, RTE = Zeitfehler
IN 2	Option SHIR-Auswahl	MDM	RPLK	LOD	SPRSW	ID3	ID2	ID1	ID0	MDM = Modem, KBLK = Tastatursperre, LCD = unterer Tastenwert freigegeben, SPRSW = Reserve, ID0-ID4 = Terminal-Identifikation
IN 3	Tastatur I/F	KE7	KE6	KE5	KE4	KE3	KE2	KE1	KE0	Tastaturdaten
IN 6	Interruptlogik	INT	1	1	1	IA3	IA2	IA1	IA0	INT = Interruptanforderung unbearbeitet, IA1-IA3 = Interruptadresse

KAPITEL 13

DER MIKROPROZESSOR IN AVIONIKSYSTEMEN

In diesem Kapitel werden zuerst die allgemeinen Probleme behandelt, um dann unter Berücksichtigung der Möglichkeiten des Mikroprozessors nach entsprechenden Lösungen zu suchen. Anschließend werden die für ein typisches Anwendungsbeispiel erforderliche Hard- und Software behandelt.

Der Mikroprozessor in Avionik-systemen

Dieses Kapitel befaßt sich mit den Avionik-Anforderungen, mit denen der Mikroprozessor konfrontiert wird. Dieser Bereich stellt ganz besondere Anforderungen an einen Mikroprozessor; andererseits können gerade Avioniksysteme einen hohen Nutzen aus der Anwendung der Mikroprozessor-Hardware- und Software-Technologie ziehen.

Die generellen Bereiche der Avionik, die in diesem Kapitel behandelt werden, sind Avionik-Funktionen, Arten der Avionik-Verarbeitung, die Rolle eines Avionik-Mikroprozessors und Lösungswege für den Hardware- und Software-Entwurf.

Der Ausdruck Avionik setzt sich zusammen aus den Wörtern "**Avia**tion" (Luftfahrt) und Elektr**onik**. Beispiele für Avionik-Geräte sind Radar, Höhenmesser, Steuerkonsolen, Displays und Rechner.

Avionik-Funktionen
(Bild 13.1)

Die durch die verschiedenen Avionik-Geräte gegebenen Avionik-Funktionen sind in Bild 13.1 zusammengefaßt.

System-Komponenten
(Bild 13.2)

Jedes System besteht aus drei Grundeinheiten: Aufnehmer (Sensoren) oder Eingabe-Prozessor und Antriebe oder Ausgabe. Sie sind in Bild 13.2 dargestellt. Die Eingangssignale für das System werden von verschiedenen Sensor- bzw. Aufnehmer-Elementen abgeleitet. Solche Sensoren sind z.B. der Steuerknüppel des Piloten, Temperaturaufnehmer und Synchro- oder Analogelemente, deren Signale den Sinus oder Kosinus eines Winkels darstellen. Diese Informationen werden auf verschiedene Arten verarbeitet – sie können geprüft, in Logiksignale umgesetzt, für arithmetische Berechnungen benützt oder lediglich einem Relais am Ausgang zugeführt werden. Die Ausgabe in einem Avioniksystem kann ein Display, ein Sensor- oder Antriebselement, oder eventuell andere Avionik-Untergruppen oder -Funktionen sein.

Architektur des Avioniksystems F-4E
(Bild 13.3)

Alle Avionik-Geräte sind mit irgendeiner Verarbeitungsaufgabe betraut. Ursprünglich wurde die gesamte Verarbeitung von speziell dafür entworfenen Analogschaltungen ausgeführt. Ein Beispiel ist in Bild 13.3 dargestellt.

- Navigation
- Flugkontrolle
- Waffenleitsystem
- Kraftstoffaufbereitung
- Triebwerk-Lufteinlaßsteuerung
- Informationsdarstellung
- Cockpit-Überwachung

Bild 13.1
Avionik-Funktionen

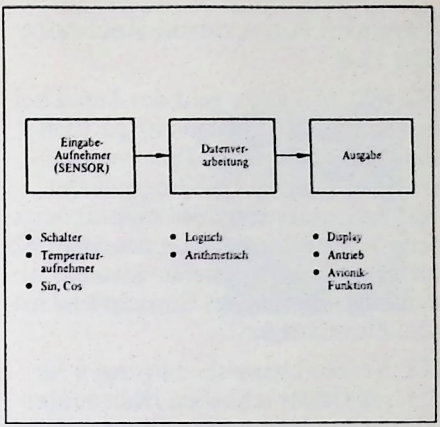


Bild 13.2
System-Komponenten

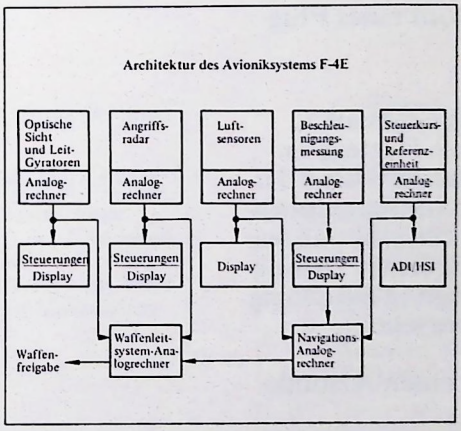


Bild 13.3
Architektur des Avioniksystems F-4E

Vergleich von Avionik-Rechnern
(Bild 13.4)

Wie Bild 13.4 zeigt, geht der Trend bei den heutigen Flugzeugen in die Richtung digitaler Avionik-Datenverarbeitung. Die digitale Ausführung bietet dem Anwender verschiedene grundsätzliche Vorteile hinsichtlich der Architektur seines Avioniksystems: kleinere Abmessungen, geringeres Gewicht und höhere Zuverlässigkeit.

Die Betriebs-Umweltbedingungen für Avionik-Geräte schließen Höhenunterschiede von Meeresniveau bis zu 24 000 m und Temperaturbereiche von −55 bis +100 °C ein. Für die Bauelemente im Innern des Gerätes bedeutet 100 °C Betriebstemperatur, daß sie Temperaturen bis zu +125 °C ausgesetzt sind.

Ausrüstungen an Bord eines Flugzeuges
(Bild 13.5)

Bild 13.5 zeigt, daß im Gegensatz zu Boden-Ausrüstungen ein Flugzeug ein vollkommen selbstständiges System darstellt, das seine eigene Stromerzeugungsanlage, sein eigenes Kühlsystem und sein eigenes Antriebssystem besitzt. Deshalb muß jedes Teil der Flugzeug-Ausrüstung entsprechend entworfen sein.

Anforderungen an einen Avionik-Rechner
(Bild 13.6)

In Bild 13.6 sind einige funktionsmäßige Voraussetzungen für Berechnungen im Avionikbereich zusammengefaßt. Die Avionik-Datenverarbeitungssignale entsprechen denen von Bodenstationen. Manche Daten werden codiert, z.B. in den BCD-Code, andere Variable werden in verkettete Felder von Wörtern zusammengefaßt.

Avionik-Berechnungen
(Bild 13.7)

Die für die Avionik-Datenverarbeitung notwendigen Berechnungen sind durch die in Bild 13.7 angegebenen Punkte charakterisiert. Diese Operationen müssen in sehr kurzer Zeit und mit einem Minimum an Speicherkapazität ausgeführt werden. Außerdem muß auf irgendeine Weise gewährleistet sein, daß keine Programm- und lebenswichtige Daten verlorengehen. Übliche Methoden zur Realisierung von nichtflüchtigen Speichern schließt Kernspeicher, Batterie-Stromversorgung, programmierbare Festwertspeicher (PROM) oder Festwertspeicher (ROM) ein.

Vergleich von Avionik-Computern

System	F-4E	F-15
Merkmale	Analog	Digital
Abmessungen (cm)	43 x 30 x 19	30 x 16 x 17
Gewicht (kg)	18,14	7,85
Stromversorgung	115/200 V 400 Hz/125 VA	115 V/400 Hz/85 VA

Bild 13.4
Vergleich von Avionik-Computern

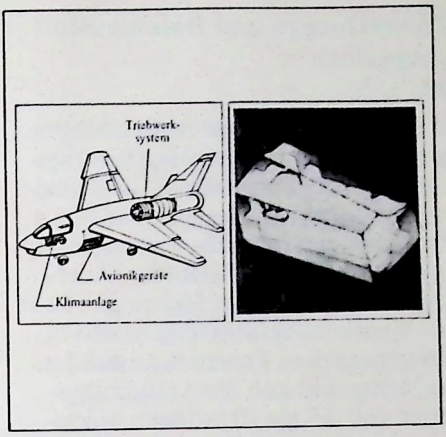


Bild 13.5
Flugzeug-Ausrüstungen an Bord

Anforderungen an einen Avionik-Rechner
Datenarten

Signal	Funktion	Dig. Darstellung
Diskrete	Ein/Aus-Si-	Ein-Bit binär
Synchro	gnale	10 ÷ 12-Bit binär
Analog	Sin/Cos eines	mit Vorzeichen
	Winkels	10 ÷ 14-Bit binär
	Druckwerte	mit Vorzeichen

Bild 13.6
Anforderungen an einen Avionik-Rechner

Avionik-Berechnungen
Merkmale

- Entscheidungen treffen
- Logische Funktionen:
  UND, ODER, Exklusiv-ODER, Komplement
- Daten komprimieren/entschlüsseln
- Arithmetische Funktionen:
  Addieren, Subtrahieren, Multiplizieren, Dividieren

Bild 13.7
Avionik-Berechnungen

Anwendungen und Befehlausführungszeiten
(Bild 13.8)

In Bild 13.8 sind einige Anwendungen zusammen mit den Befehlausführungszeiten grafisch aufgetragen. Die Signalverarbeitung ist für Systeme wie Sonar, Radar und elektronische Kriegsführungssensoren typisch und erfordert Ausführungszeiten von 10 bis 500 ns. Bei Echtzeit-Verarbeitungsfunktionen, also bei Navigations- und Feuerleit-Anwendungen, erstrecken sich die Ausführungszeiten von 0,5 bis 10 μs. Beim Mikroprozessor liegen die Befehls-Ausführungszeiten zwischen 2 und 100 μs; er ist damit teilweise für Echtzeit- als auch für andere kritische Anwendungen geeignet, wie beispielsweise zur Steuerung von Geräten oder Ablaufsteuerungen.

Avionik-Mikroprozessoranwendungen
(Bild 13.9)

Es gibt mehrere potentielle Avionik-Bereiche, in denen man Mikroprozessoren einsetzen kann. Sie sind in Bild 13.9 angegeben. Zwei mögliche Anwendungen sind Geräte-Funktionstests und die Datenaufbereitung, um sie auf Displays darstellen oder anderen Systemteilen zuführen zu können. Nahezu jede Avionik-Funktion mit mittlerer Verarbeitungsgeschwindigkeit und begrenzten Speicher-Anforderungen ist ein Kandidat für Mikroprozessor-Anwendungen. Dies schließt z.B. Funktionen wie Flugdaten-Berechnungen, Triebwerk-Luft-einlaßsteuerung, Radio-Navigation und digitale Autopiloten ein.

Prozessor-Datenformat
(Bild 13.10)

Um die Anwendung der Mikroprozessor-Technologie auf dem Avionikgebiet zu demonstrieren, gehen wir von einem System aus, das Daten von einem Sensor empfängt, aus diesen Daten die augenblickliche Flugzeugposition errechnet und schließlich diese Daten für den Piloten auf einem Display darstellt. Zur besseren Übersicht wollen wir uns auf den folgenden Teil des Verarbeitungsprozesses konzentrieren: Lesen der Daten vom Sensor, Entschlüsseln und Umsetzen der Daten in eine geeignete Form, Plausibilitäts-Prüfung des Ergebnisses und Speichern der gültigen Daten für die erneute Positionsberechnung. Wie in Bild 13.10 gezeigt, liegen die Eingangsdaten in einem komprimierten 16-Bit-Format vor. Die 14 niedrigstwertigen Bits beinhalten eine im Zweier-Kompliment-Code codierte Zahl, die den Sinus oder Kosinus eines Winkels darstellen. Die zwei höchstwertigen Bits sind Hardware-Statusbits.

Flußdiagramm zur Datenverarbeitung
(Bild 13.11)

Bild 13.11 zeigt, daß für die Verarbeitung folgende Schritte erforderlich sind: Ausblenden der unnötigen Bits, Kopieren des Vorzeichens zur Bildung einer 16-Bit-Zahl, Berechnung der Summe der Quadrate des Sinus- und Kosinuswertes und Prüfung des Ergebnisses, ob es innerhalb des erlaubten Wertebereichs liegt. Ist das Ergebnis plausibel, wird von dieser Variablen die Quadratwurzel gezogen; danach werden zwei neue Ausdrücke berechnet. Ist das Ergebnis nicht korrekt, wird zu einer Fehler-Verarbeitungsroutine verzweigt.

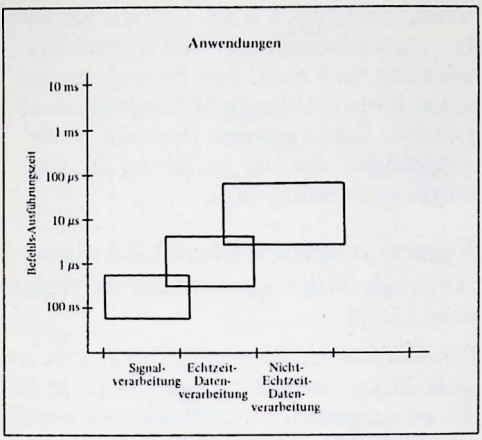


Bild 13.8
Anwendungen und Befehls-Ausführungszeiten

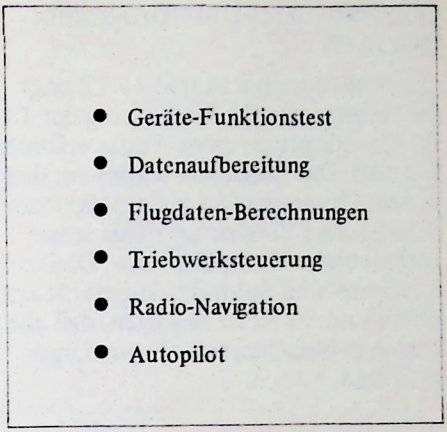


Bild 13.9
Avionik-Mikroprozessoranwendungen

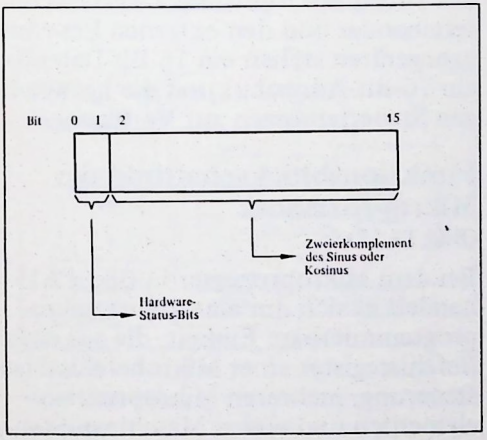


Bild 13.10
Prozessor-Datenformat

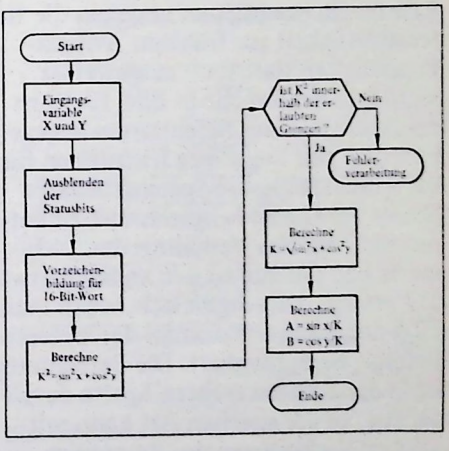


Bild 13.11
Datenverarbeitungs-Flußdiagramm

Quadratwurzel-Unterprogramm
(Bild 13.12)

Das Flußdiagramm in Bild 13.12 zeigt die Schritte, die zur Berechnung der 16-Bit-Quadratwurzel eines Wertes erforderlich sind. Das Diagramm deutet an, daß in dem Quadratwurzel-Unterprogramm Schiebe- und Vergleichsbefehle sowie arithmetische Multiplikations-, Divisions-, Additions- und Subtraktionsbefehle enthalten sind. Es ist zu beachten, daß alle Zahlen in hexadezimaler Form dargestellt sind.

Befehlshäufigkeit
(Bild 13.13)

Der erforderliche Speicher für ein solches Problem läßt sich rasch abschätzen, indem man ähnliche Anwendungen untersucht. Wenn derartige Informationen nicht verfügbar sind, kann man sich an Richtlinien orientieren, in denen die Befehlshäufigkeit aus früheren Avionik-Programmen statistisch ausgewertet wurde. In der Tabelle in Bild 13.13 ist die Häufigkeit der Befehlsarten angegeben; und gibt ungefähre Richtlinien für die Abschätzung des Speicherbedarfs. Dieser Tabelle ist beispielsweise zu entnehmen, daß das Verhältnis der arithmetischen Befehle zu den anderen etwa 1:7 beträgt. Dies ergibt sich, wenn man 12,5 durch die Gesamtzahl der anderen Befehle, 87,5, dividiert. Die Berechnung ist in der äußeren rechten Spalte durchgeführt. In der gleichen Art kann mit solchen Richtwerten der Programmspeicher und Datenspeicherbedarf für eine gegebene Anwendung abgeschätzt werden.

Um die System-Anforderungen zu erfüllen, wird ein 16-Bit-Mikroprozessor-System mit PROM-Programm- und Konstanten-Speicher und RAM-Datenspeicher benötigt. Das System darf nur wenig Leistung verbrauchen, muß klein sein und innerhalb eines weiten Temperaturbereichs arbeiten. Unter diesen Voraussetzungen muß sich die Hardware-Entwurfslösung nach den Systemeigenschaften und nach den Erfordernissen eines 16-Bit-Avionik-Mikroprozessors richten. Dann werden jene Bausteine ausgesucht, die zur Erfüllung des Entwurfs notwendig sind.

Funktionsblockschaltbild eines Avionik-Mikroprozessor-Systems
(Bild 13.14)

Ein Funktionsblockschaltbild eines Avionik-Mikroprozessor-Systems ist in Bild 13.14 dargestellt. Der Prozessor besteht aus einem 16-Bit-Mikroprozessor mit Takt und Steuerung, einem Programm-/Konstantenspeicher (ROM) und einem schnellen Daten-Zwischenspeicher-(scratch-pad memory) RAM. Für den Datenverkehr und die Datensteuerung zwischen den drei Funktionselementen des 16-Bit-Mikroprozessor-Systems untereinander und den externen Ein-/Ausgabegeräten stehen ein 16-Bit-Datenbus, ein 16-Bit-Adressbus und die notwendigen Steuerleitungen zur Verfügung.

Funktionsblockschaltbild des Mikroprozessors
(Bild 13.15)

Bei dem Mikroprozessor in Bild 13.15 handelt es sich um eine kleine mikroprogrammierbare Einheit, die aus einem Befehlsregister einer Mikrobefehlsablaufsteuerung, mehreren Mikroprozessorelementen und einem Maschinen-Statusregister besteht.

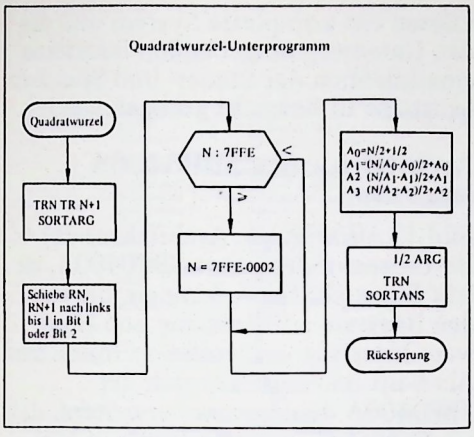


Bild 13.12
Quadratwurzel-Unterprogramm

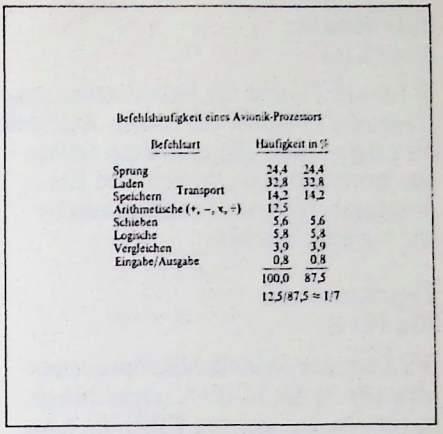


Bild 13.13
Prozessor-Befehlsbenützung

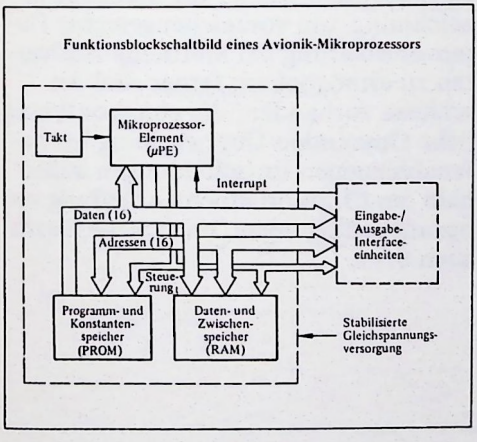


Bild 13.14
Funktionsblockschaltbild eines Avionik-Mikroprozessors

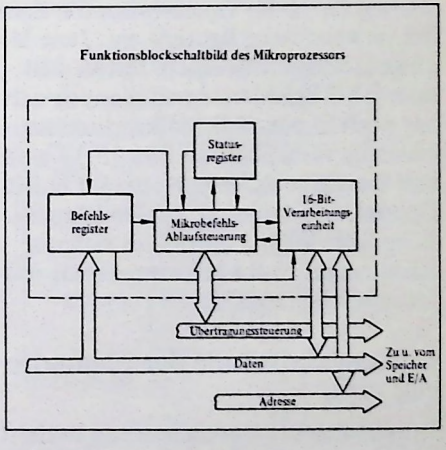


Bild 13.15
Blockschaltung des Mikroprozessor-Elements

Befehlsarten
(Bild 13.16)

In Bild 13.16 sind die Befehlsarten angegeben, die mit einer derartigen Architektur möglich sind. Sie enthalten Datentransport, Schiebe-, Steuer- und Ein-/Ausgangsbefehle sowie arithmetische und logische Befehle.

Typenauswahl
(Bild 13.17)

Der Kern des Avionik-Mikroprozessorentwurfs ist die 16-Bit-Verarbeitungseinheit. In den meisten Fällen ist eine funktionelle Systemarchitektur ein Spiegelbild der erhältlichen Bausteine und deren Leistungsfähigkeit. Der weite Arbeitstemperaturbereich beschränkt die Auswahl vornehmlich auf bipolare Bauelemente. Weiterhin engt die Voraussetzung für 16-Bit-Operationen die Zahl der verwendbaren Bauteile ein. Jene Mikroprozessoren mit einem festen 4-Bit- oder 8-Bit-Befehlssatz entfallen, so daß nur noch 2- oder 4-Bit-Mikroprozessorelemente verbleiben. In Bild 13.17 sind drei Bausteine aufgeführt, die für N-Bit-Operationen ausgelegt werden können. Beim SBP9900MJ handelt es sich um einen neuen 16-Bit Mikroprozessor mit festem Befehlssatz.

Leistungsaufnahme der Elemente
(Bild 13.18)

Hinsichtlich der Verlustleistung besteht zwischen den vier Bausteinen ein offensichtlicher Unterschied. Wenn man die Avionik-Kriterien "geringe Leistung, minimale Kühlung" und die in Bild 13.18 angegebenen Daten betrachtet, fällt die Wahl auf den SBP0400AM oder den SBP9900MJ. Dies wird noch deutlicher, wenn man von diesen vier Bausteinen – ausgelegt für einen 16-Bit-Datenweg – kurz die Verlustleistung berechnet. Bevor allerdings eine vollständige und endgültige Verlustleistungs-Berechnung durchgeführt werden kann, müssen das komplette System und die den Datenweg umgebenden Bausteine einschließlich der Steuer- und Speicherbausteine in Betracht gezogen werden.

Architektur des SBP0400A
(Bild 13.19)

Bild 13.19 zeigt die Architektur des 4-Bit-Prozessorelementes SBP0400A, sie läßt Eigenschaften erkennen, die für den Baustein auf Leistung und Hardware-Interface von großem Einfluß sind. Als 4-Bit-Baustein läßt sich der SBP0400A dahingehend erweitern, daß ein Datenweg variabler Breite in Vielfachen von 4 Bits realisiert werden kann. Weitere Merkmale zur Leistungsverbesserung sind eine vorausschauende Befehlseinrichtung zur Überlappung von Befehlsabruf und Ausführungszyklus, eigene Anschlüsse für die Stellenkennzeichnung, um vorzeichengerechte Datenverarbeitung bei Multichip-Aufbauten zu ermöglichen; ferner sind Anschlüsse vorhanden, die eine kontinuierliche Operanden-Überwachung bei datenabhängigen Entscheidungen außerhalb des Chips erlauben (Ableitung von Sprungbedingungen, etc.) Siehe hierzu auch 6.11.

Datentransport
- Laden
- Speichern

Arithmetische
- Addieren
- Subtrahieren
- Vergleichen
- Multiplizieren
- Dividieren

Logische
- Laden Einer-Komplement
- UND
- ODER
- Exklusiv-ODER

Schieben
- Schieben 1 Position
- Schieben 2 Positionen

Programm- und Interruptsteuerung
- Bedingt springen
- Vertauschte Status & PC
- Setze Status
- Schleifensteuerung
- Unterprogramm-Verschachtelung

Eingabe/Ausgabe
- Speicher-Eingabe
- Speicher-Ausgabe

Bild 13.16
Befehlsarten

Mögliche Typen für Avionik-Anwendungen

Typenbezeichnung	Technologie	Zahl der erford. Bausteine für einen 16-Bit-µP
SBP0400AM	Bipolar-I^2L	4
SN54S481J	Bipolar-Schottky	4
3002	Bipolar-Schottky	8
SBP9900MJ	Bipolar-I^2L	1

Bild 13.17
Typenauswahl

Leistungsaufnahme der Elemente

Typ	Max. Verlustleistung (Watt)	Gesamte Verlustleistung (Watt)
0400A	0,145* +.71	0,58* + 2,84
S481	1,40	5,60
3002	0,788	6,30
9900	0,5* + 2,0	2,50

* Enthält nicht die am Stromversorgungs-Vorwiderstand verbrauchte Leistung.

Bild 13.18
Leistungsaufnahme der Elemente

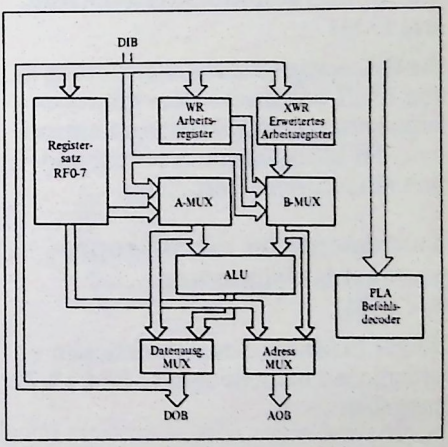


Bild 13.19
SBP0400A-Architektur

SBP0400A-Befehlsformate
(Bild 13.20)

Für dieses Beispiel wird ein 16-Bit Mikroprozessor mit mehreren SBP0400A und den daraus resultierenden Eigenschaften aufgebaut. Bild 13.20 zeigt dessen sechs möglichen Befehlsklassen. Eine Untersuchung dieser sechs Klassen im Anhang liefert genauere Hinweise auf die Eigenschaften eines Systems mit dem SBP0400A. Ausgehend von diesem Mikrobefehlssatz und dem vorhandenen Registersatz von 8 Registern kann der Aufbau eines 16-Bit-Mikroprozessors vorgenommen werden. Für dieses Beispiel wird für jedes Funktionselement des 16-Bit-Systems ein Satz Bausteine mit möglichst kleiner Verlustleistung ausgewählt.

Mikroprogrammablaufsteuerung
(Bild 13.21)

Die Mikroprogrammablaufsteuerung in Bild 13.21 besteht aus dem Mikrobefehlsregister, dem Mikroprogrammspeicher, der Mikrobefehlssteuerung und dem Mikrobefehls-Decodierer.

Bauelemente für die Mikroprogrammablaufsteuerung
(Bild 13.22)

Die zur Erfüllung dieser Funktionen erforderlichen Bauteile sind in Bild 13.22 angegeben.

Bauelemente-Anzahl und Leistungsaufnahme
(Bild 13.23)

Bild 13.23 zeigt den endgültigen Entwurf und die ausgewählten Bauelemente. Der Mikroprozessor besteht aus etwa 48 integrierten Bausteinen und verbraucht ungefähr 12,2 Watt.

Befehlsformat	Befehlsart
I	A(ALU)B → A or B or C
II	A plus B plus ALUCIN → A or B or C
III	A → B
IV	(WR +/−A plus ALUCIN) SHIFTED → WR
V	(WR plus ALUCIN) SHIFTED → WR
VI	(WR plus ALUCIN, XWR) SHIFTED → (WR, XWR)

Bild 13.20
SBP0400A-Befehlsformate

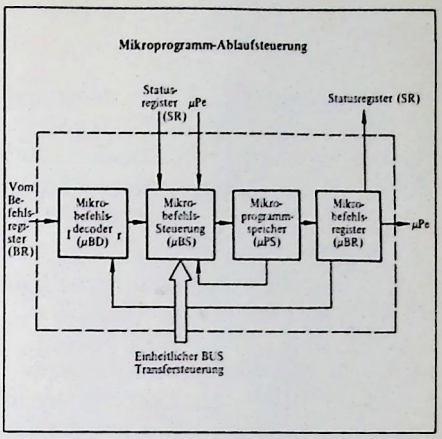


Bild 13.21
Mikroprogrammablaufsteuerung

Mikrobefehlsregister	54LS174J
Mikroprogrammspeicher	54S387
Mikrobefehlssteuerung	54LS174J
	54LS153J
	54LS11J
	54LS83J
	54LS151J
	54LS163J
Mikrobefehlsdecoder	54LS157J
	54S387
Befehlsregister	54LS174J
	54LS175J
	54LS163J
	54LS157J
Statusregister	54LS174J
Hilfslogik	54LS00J
	54LS74J
	54LS365J
Takt (CLK)	54LS124J
Speicher	
Programm 2k x 16 (PROM)	54S471J
Daten (256 x 16 (RAM)	54LS201J

Bild 13.22
Bauelemente der Mikroprogrammablaufsteuerung

Bauelemente-Anzahl-Leistungsaufnahme

16-Bit-Mikroprozessor	Zahl der Bausteine	Max. Leistungsaufnahme
µPe	4	58
µBR	4	32
µPS	6	3·90
µBS	11	55
µBD	4	1·30
BR	5	37
SR	1	08
Hilfslogik + Takt	13	2·27
Diskrete Bauelemente	–	2·84
	48	12·21

Bild 13.23
Bauelemente-Anzahl und Leistungsaufnahme

Anzahl der Speicher
(Bild 13.24)

Der Speicher umfaßt 35 Bausteine und nimmt eine Leistung von 7,5 W auf, wenn vorausgesetzt wird, daß Schaltung der Versorgungsspannung der PROM's zur Reduzierung der Verlustleistung durchgeführt wird.

Natürlich werden noch zusätzliche Bausteine für bestimmte Ein-/Ausgabe-Schnittstellen usw. benötigt. Jedoch schließt diese Leistungsverbrauchsangabe den Vorwiderstand mit ein, der in Serie mit U_{CC} verbunden wird, um den erforderlichen Injektorstrom von 150 mA für jeden SBP0400-Baustein zu erzeugen.

Mikroprozessor-Layout
(Bild 13.25)

Anhand von Bild 13.25 kann man abschätzen, wieviel Leiterplattenfläche man für die integrierten Schaltungen des Mikroprozessor-Systems benötigt. Hierzu berechnet man die Fläche der verschiedenen Gehäusegrößen und ermittelt dann die Gesamtfläche für eine Multilayer-Leiterplatte. Ähnliche überschlägige Berechnungen kann man für die von den Speichern belegte Fläche vornehmen. Schließlich muß man noch etwas Platz freihalten für diskrete Bauelemente, wie Entkoppelkondensatoren, Pull-up-Widerstände usw. Alles in allem würden die 48 Gehäuse des vollständigen 16-Bit-Mikroprozessor-Systems auf einer Multilayer-Leiterplatte eine Fläche von etwa 150 cm^2 beanspruchen.

Software-Entwurf und -Entwicklung
(Bild 13.26)

Wie bei der Hardware gibt es auch beim Software-Entwurf mehrere einzigartige Voraussetzungen, nach denen man sich bei der Avionik-Software-Entwicklung zu richten hat. Die Forderungen nach wohldurchdachter Software beziehen sich auf Größe, Gewicht, Verlustleistung, Takt- und Speicher-Anforderungen, Genauigkeit und Flugsicherheit.

Die Erprobung der Avionik-Software, im Labor oder während des Fluges, mit Hilfe von Flugzeug-Festanlagen kann sehr teuer sein. Deshalb sollte man zweckmäßigerweise Simulations-Techniken anwenden, um die Software zu überprüfen und zu verifizieren.

Die Anfangsgröße des Speichers und die Anforderungen an die Ausführungszeiten sind kritische Bereiche für den Software-Entwurf. Alle Abschätzungen sollten auf einer genauen Beschreibung der Systemfunktionsweise und der Ein-/Ausgangs-Datenübertragungsrate beruhen. Jedes Projekt unterscheidet sich in der Komplexität der Anwendung, aber dennoch gibt es einige grundsätzliche Betrachtungen zur Avionik-Software-Entwicklung. Sie sind in Bild 13.26 aufgeführt.

Zusammenfassung

Dieses Kapitel befaßte sich mit den System-Voraussetzungen für den Einsatz eines Avionik-Mikroprozessors. Es erläuterte ferner einige gebrauchsfähige Hardware- und Software-Entwürfe. Die wesentlichen Vorteile von Mikroprozessoren in Avionik-Systemen sind reduzierte Kosten, Größe, Gewicht und Leistungsverbrauch, in Verbindung mit höherer Flexibilität hinsichtlich Programmierbarkeit und Software-Modifikationen.

Speicher

Programm- und Konstantenspeicher (2k x 16 PROM)	16	2,6*
Datenspeicher (256 x 16)	16	4,4
Zusätzl. Elemente	3	0,5
Summe	35	7,5

* einschließlich Reduzierung durch Schalten der PROM-Versorgungsspannung

Bild 13.24
Anzahl der Speicher

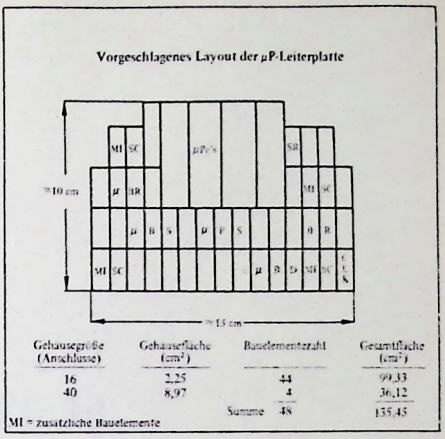


Bild 13.25
Mikroprozessor-Layout

- Logikmodell in höherer Programmiersprache, wie z.B. FORTRAN
- Programmierung in Mikroprozessor-Assemblersprache
- Simulation auf Mikrobefehlsebene
- Simulation der Ein-/Ausgabe-Umgebungsbedingungen

Bild 13.26
Software-Entwurf und -Entwicklung

KAPITEL 14

MIKROPROZESSOREN IN AUTOMOBILEN

Dieses Kapitel beschäftigt sich mit den Systemen von Kraftfahrzeugen, in denen Mikroprozessoren praktisch eingesetzt werden können. Nach der Darstellung und Untersuchung der typischen Probleme wird erläutert, wie diese durch Mikroprozessoren gelöst werden können.

Mikroprozessoren in Automobilen

Dieses Kapitel untersucht die System-Bedingungen von Kraftfahrzeug-Anwendungen und gibt einen Überblick über die Mikroprozessor-Parameter, die zur Erfüllung der Kfz-Systemanforderungen notwendig sind. Außerdem wird beschrieben, wie ein Mikroprozessor als typische Automobil-Steuereinheit verwendet werden kann.

Nicht alle Autos sind gleich
(Bild 14.1)

Der Automobil-Markt und die -Produkte sind in verschiedener Hinsicht einzigartig. Jedes vom Markt akzeptierte Produkt wird in großen Stückzahlen hergestellt. Wenn nun schon die Automobile in solch großen Stückzahlen gefertigt werden, wäre ein Standard-Produkt das Beste. In Wirklichkeit herrscht jedoch, wie Bild 14.1 illustriert, auf dem Automobil-Markt eine starke Abneigung gegen jegliche Standardisierung.

Aufgrund seiner Vielseitigkeit durch Änderung der Software bietet der Mikroprozessor die Möglichkeit, in unterschiedlichen Modellen eingesetzt zu werden. Da ein Mikroprozessor als Standard-Bauelement in vielen verschiedenen Automodellen verwendbar ist, kann er in der Tat die beste Lösung für diese Probleme darstellen.

Bei einem Kraftfahrzeug stehen Sicherheit und Zuverlässigkeit an erster Stelle. In den meisten anderen Bereichen wird versucht, Sicherheit und Zuverlässigkeit sozusagen in das Produkt "hineinzuentwickeln". Auf dem Automobil-Sektor werden in vielen Fällen solche Sicherheits- und Zuverlässigkeits-Standards häufig durch Gesetz vorgegeben. Die Nichteinhaltung dieser Standards führt oft zu strengen Strafen, deren Auswirkung zum Beispiel darin besteht, daß Automodelle öffentlich zurückgerufen werden, was in letzter Zeit nicht selten vorkam. Es ist deshalb äußerst wichtig, diese auferlegten Sicherheits- und Zuverlässigkeits-Bestimmungen bereits beim Entwurf zu berücksichtigen.

Bild 14.1
Nicht alle Autos sind gleich

Verbesserungsmöglichkeiten durch den Mikroprozessor im Kraftfahrzeug
(Bild 14.2)

Es gibt mehrere, weit gefaßte Bereiche bei Kfz-Produkten, in denen der Mikroprozessor zu Verbesserungen führen könnte. Diese Bereiche sind in Bild 14.2 angegeben. Hiervon ist der Bereich "Sicherheit" der wichtigste. Die Betrachtung eines Kraftfahrzeuges und seiner Betriebsumwelt läßt auf zahlreiche Gebiete schließen, in denen automatische Steuerungen oder sehr komplexe Sensoren und programmierte Mikroprozessorensysteme eingesetzt werden könnten, um die Sicherheit des Produkts zu erhöhen.

Der zweite Bereich, der sicherlich einer intensiven Betrachtung unterzogen werden muß, ist der Wirkungsgrad. Es ist nicht schwierig, anhand einfacher Berechnungen auch den besonders voreingenommenen Beobachter davon zu überzeugen, daß bei den heutigen hohen Kraftstoffpreisen eine Optimierung des Wirkungsgrades von erheblicher Bedeutung ist.

Die Abgasentgiftung ist ein weiterer Bereich. Hier dürfte wohl eine in jedem Auto befindliche adaptive Regelung die bestmögliche Lösung sein, um die Emission auf ein Minimum zu beschränken.

Ein weiterer Bereich, die Diagnose, deutet darauf hin, daß es sich bei einem Produkt, das seinen Besitzer auf eventuelle Defekte oder auf einen notwendigen Service hinweist, um ein hochwertiges Produkt handelt.

Bisher sind Verbesserungen in Automobilen ausschließlich auf klassische Weise vorgenommen worden. Dabei wurden jahrelang immer nur einzelne Teile eines Gesamtsystems berücksichtigt und entsprechend dem technischen Fortschritt verbessert. Die Änderungen des Scheinwerfer-Design von Kraftfahrzeugen sind dafür beispielhafte Belege.

Automobil-Systembetrachtungen
(Bild 14.3)

Im Gegensatz zu diesem Entwurfsprinzip sind die Entwurfskonzepte mit einem Mikroprozessor erheblich systemorientierter, d.h. mit ihnen können große Teile eines Systems verbessert werden, wie aus Bild 14.3 hervorgeht. Deshalb beschäftigt sich dieses Kapitel zuerst mit dem Gesamtsystem, um danach die verschiedenen Systeme eines Kraftfahrzeugs in Bereiche aufzugliedern, in denen durch Verwendung eines Mikroprozessors Verbesserungen erzielt werden können.

Eine Untersuchung der Umweltbedingungen, unter denen ein System betriebsfähig sein muß, ist deshalb wichtig, weil das Auto Umwelteinflüssen ausgesetzt ist, die teilweise erheblich kritischer sind als bei den bisher behandelten Geräten. Anhand eines dieser Systeme wird erläutert, wie Mikroprozessoren eingesetzt werden können und welche Parameter sie aufweisen müssen, damit es als eine effektive Lösung betrachtet werden kann.

Darüber hinaus werden potentielle Software-Anforderungen und Speichergröße für eine Automobil-Anwendung eines Mikroprozessors diskutiert.

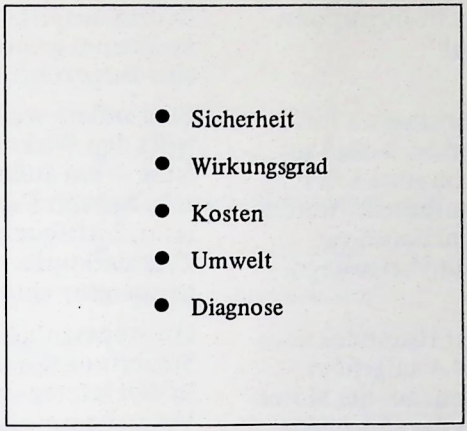


Bild 14.2
Verbesserungsmöglichkeiten durch den Mikroprozessor in Kraftfahrzeugen

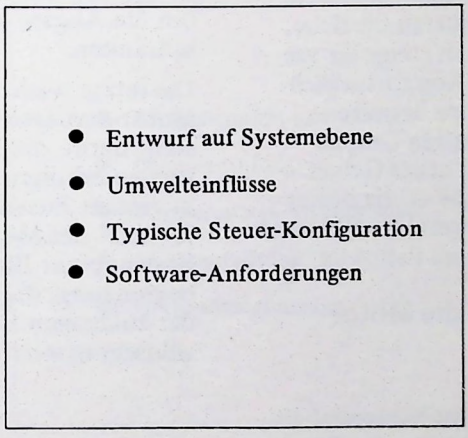


Bild 14.3
Automobil-Systembetrachtungen

Die fünf Hauptfunktionsgruppen in einem Automobil
(Bild 14.4)

Anstatt nach einer schrittweisen Entwurfsmethode vorzugehen, wollen wir hier die Gesamtfunktion eines Kfz's untersuchen. Darauf aufbauend, wurden einige weit gefaßte, funktionell verwandte Systeme im Auto herausgegriffen.

Es gibt mindestens fünf Hauptfunktionsgruppen, die in Bild 14.4 aufgeführt sind. Sie bestehen erstens aus der Motor-Steuereinheit, die alles das einschließt, was ausschließlich mit der Energieerzeugung zusammenhängt; zweitens aus der Getriebe-Steuereinheit, die – primär zum Zwecke des Antriebs – wirkungsvollen Gebrauch von der vom Motor abgegebenen Leistung macht. Die dritte Gruppe, die Sicherheitseinrichtungen, berührt nahezu alle anderen Bereiche; hat aber zweifellos auch einige ihr spezifische Aspekte. Die Komforteinrichtungen als vierte Gruppe betreffen den Luxus. Bei der fünften Gruppe handelt es sich um ein neues Gebiet – die Diagnoseprogramme –, die bisher noch nicht in diesem Bereich erschienen sind, zumindest nicht im Fahrzeug.

Anforderungen an die Motor-Steuereinheit
(Bild 14.5)

Die Aufgaben der Motor-Steuereinheit sind in Bild 14.5 zusammengestellt. Eine Motorsteuerung sollte den Zündzeitpunkt einstellen. Ferner könnte sie auch die Höhe der Zündfunken-Energie bei unterschiedlichen Betriebsbedingungen regulieren.

Darüber hinaus sollte eine Motor-Steuereinheit die Benzinzufuhr steuern, wobei – entsprechend dem Trend der letzten Jahre – vielleicht eine programmierte Benzineinspritzung die wirkungsvollste Methode sein könnte. Für eine solche Benzineinspritzung ist ein von einem Programm gesteuerter Mikroprozessor eine ausgezeichnete Lösung.

Eine andere wichtige Steuerfunktion betrifft den Wirkungsgrad der Verbrennung – ein äußerst vielschichtiger Prozeß, der von Faktoren wie Lufttemperatur, Luftfeuchtigkeit, Einlaßdruck, Zylinderkopftemperatur und Abgastemperatur abhängt.

Die Abgasentgiftung stellt eine weitere Steuerfunktion der Motorsteuerung dar. In den letzten Jahren sind verschiedene Versuche unternommen worden, um die Abgase zu reduzieren. Ein Mikroprozessor kann hier über eine adaptive Regelung Zusätzliches leisten, wobei das Fahrzeug ohne weiteres in der Art modifiziert werden kann, daß es sich an die jeweils vorherrschenden Umgebungs- und Betriebseinflüsse anpaßt, um die Abgase auf ein Minimum zu beschränken.

Die letzte wichtige Aufgabe, die eine Motor-Steuereinheit zu erfüllen hat, besteht darin, die Lebensdauer eines Motors zu erhöhen. Ein kritischer Faktor in diesem Zusammenhang ist das "Überdrehen" des Motors. Deshalb ist der Einbau einer Einrichtung zur Drehzahlbegrenzung, die das Fahrzeug innerhalb der zulässigen Betriebsgrenzen hält, sehr wünschenswert.

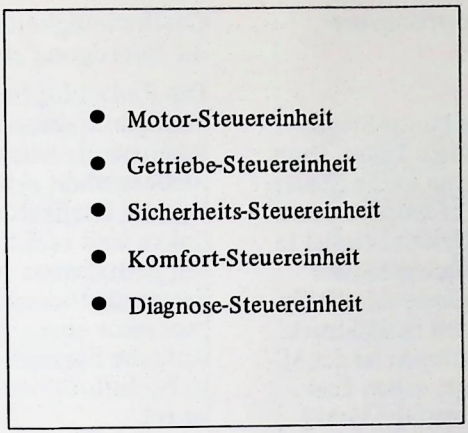


Bild 14.4
Die fünf Hauptfunktionsgruppen in einem Automobil

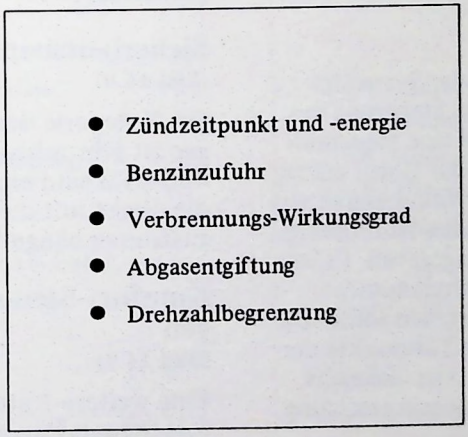


Bild 14.5
Anforderungen an die Motor-Steuereinheit

Beispiel einer Mikroprozessor-Motorsteuerung
(Bild 14.6)

Da man nun die für die Motor-Steuereinheit wichtigen Faktoren kennt, kann man damit beginnen, eine solche Steuereinheit mit Hilfe eines Mikroprozessors zu entwerfen. In dem Systembeispiel in Bild 14.6 besitzt der Mikroprozessor Eingänge für die Motordrehzahl, die Zylindertemperatur und den Einlaßdruck. Anhand dieser Informationen ist der Mikroprozessor in der Lage, neben dem Luftzufuhr- und Benzinzufuhr-Ventil ein Abgassystem zu steuern, das über eine Rückführung zum Motor zugleich die Auspuffgase reguliert. Das Interface zum nächsten Steuersystem würde sich an der Ausgangswelle befinden.

Anforderungen an die Getriebe-Steuereinheit
(Bild 14.7)

Die nächste Kategorie der Systembetrachtungen betrifft die Steuerung des Getriebes. Hierfür wird eine Steuereinheit benötigt, die die vom Motor erzeugte Energie mit hohem Wirkungsgrad ausnutzt. Die entsprechenden Anforderungen sind in Bild 14.7 angegeben. Es ist einleuchtend, daß die Drehmoment-Last-Kurve optimiert werden sollte. Daraus folgt, daß die Umschaltpunkte der Standard-Übersetzung oder vielleicht auch eine variable Getriebeübersetzung gesteuert werden.

Die Geschwindigkeitsbegrenzung ist eine weitere Aufgabe. Einige Autos (vor allem in den USA) verfügen derzeit über eine "Reisegeschwindigkeits"-Steuereinrichtung. Ein solches System läßt sich ohne weiteres in einer Mikroprozessor-Steuereinheit verwirklichen, wobei man die gewünschten Grenzwerte in ein Register eingibt. Dadurch vereinfacht sich die Geschwindigkeitsbegrenzung auf einen periodischen Vergleich des Registerinhalts mit dem Wert des Geschwindigkeitsaufnehmers und auf die Betätigung eines Alarmgebers.

Die Radschlupfbegrenzung ist bei den heutigen Fahrzeugen zweifellos wünschenswert. Auch sie könnte mit einer Steuereinheit erfolgen, indem man lediglich die Drehzahldifferenz zwischen dem linken und rechten Antriebsrad bestimmt. Mit dem daraus resultierenden Signal, dessen Aufbereitung für einen Mikroprozessor eine einfache arithmetische Aufgabe darstellt, wird das Differential so beeinflußt, daß es ohne Schlupf arbeitet.

Bei der Übertragungsdrosselung handelt es sich um einen neuen Prozeß, der bis heute noch nicht verwirklicht wurde. Hier wird per Programm bzw. Software die Getriebe-Steuereinheit modifiziert, um max. Belastungen des Fahrzeugs zu reduzieren.

Sicherheitsbetrachtungen
(Bild 14.8)

Die Kategorie der Sicherheitseinrichtungen ist sehr schwierig zu definieren. In Bild 14.8 sind einige Punkte aufgeführt, die direkt mit diesem speziellen Gebiet zusammenhängen.

Komfort-Steuersystemanforderungen
(Bild 14.9)

Eine weitere Kategorie umfaßt die Komfort-Steuersysteme. Es wäre nicht unbedingt notwendig, diese Kategorie hier aufzuführen. Aber warum sollten nicht die Möglichkeiten eines bereits im Auto vorhandenen Mikroprozessors genutzt werden. Die Kosten sind nicht erheblich. Einige mögliche Komfort-Einrichtungen sind in Bild 14.9 angegeben.

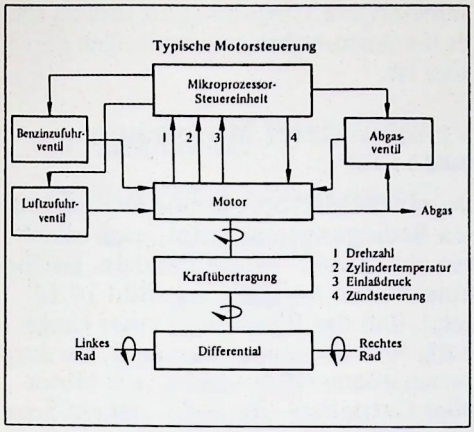


Bild 14.6
Beispiel einer Mikroprozessor-Motorsteuerung

- Optimierung von Drehmoment und Last
- Geschwindigkeitsbegrenzung
- Radschlupfbegrenzung
- Übertragungsdrosselung

Bild 14.7
Anforderungen an die Getriebe-Steuereinheit

- Antiblockiersystem
- Aufprallschutz
- System-Überwachung
- Fahrer-Zustand
- Diebstahlschutz
- Automatisches Türschloß
- Automatisches Licht

Bild 14.8
Sicherheits-Steuersystemanforderungen

- Ladeausgleich
- Federungsregulierung
- Klimaanlage
- Unterhaltungssystem
- Fahrer-Display

Bild 14.9
Komfort-Steuersystemanforderungen

Diagnose-Steuersystemanforderungen
(Bild 14.10)

Die letzte Kategorie der Systembetrachtungen im Hinblick auf die Verwendung eines Mikroprozessors im Auto betrifft die Diagnoseprogramme. Das ist für die Automobil-Hersteller ein neues, und nicht zu vernachlässigendes Gebiet. Wie in Bild 14.10 angegeben, schließen diese Betrachtungen den Bereich der Sicherheitstests ein. Damit wird beispielsweise ermittelt, ob die Bremsanlage ordnungsgemäß arbeitet.

Ein weiteres Diagnoseprogramm dient zur Überwachung des Wirkungsgrades von wichtigen Funktionen. Für einen Mikroprozessor ist es eine einfache Aufgabe, über die verschiedenen Funktionen eines Fahrzeuges zu wachen und dem Fahrer in einer vorbestimmten Weise mitzuteilen, daß er nun für 4 km 1 Liter benötigt, wo er normalerweise mit 1 Liter 10 km weit kommt.

Die letzte Kategorie der Diagnose-Steuereinrichtung beinhaltet Wartungsprogramme. Ein Wartungsprogramm könnte Subroutinen umfassen die dem Fahrer die optimale Zeit bis zum Auswechseln der Zündkerzen mitteilen würden.

Umgebungsbedingungen für Kfz-Anwendungen
(Bild 14.11)

Die Liste der empfohlenen Betriebsbereiche, in denen ein Mikroprozessor arbeiten könnte, umfaßt solche, die für die Mikroprozessor-Ausführung als angemessen erscheinen.

Bild 14.11 gibt in großen Zügen die Umgebungsbedingungen wieder, unter denen diese Systeme betriebsfähig sein müssen. Wie sich daraus entnehmen läßt, dürfte es schwierig sein, für den Betrieb eines Mikroprozessors eine schwierigere Umgebung zu finden, als es bei Automobil-Anwendungen der Fall ist.

Verschmutzter Motorraum
(Bild 14.12)

Bei einem Mikroprozessor, der unter diesen Bedingungen arbeitet, muß es sich um ein "Super"-Gerät handeln. Darüber hinaus ist es möglich, wie Bild 14.12 zeigt, daß das Element in einer Umgebung funktionsfähig sein muß, die durch verschiedene Öl-Produkte, wie Motor- oder Getriebeöl, Bremsflüssigkeit, Scheibenwasch-Lösungsmittel, Benzin, Antifrostmittel, Entfettungs- und Reinigungsmittel, Seifen, Dampf und vielleicht auch ätzende Batteriegase, verunreinigt wird. Ein Mikroprozessor kann unmöglich so gebaut werden, daß er all diesen Einflüssen widersteht. Allerdings sollten diese Bedingungen im Auge behalten werden, falls irgendein System für Kraftfahrzeuge entworfen wird.

Ein weiterer zu beachtender Faktor ist die Betriebsspannung. Bei einer Nenn-Betriebsspannung von 12 V muß man mit einer maximalen Spannung von 16 V und einer minimalen Spannung von 10 V rechnen.

Ein Mikroprozessor muß somit einer relativ schwierigen Umgebung widerstehen, die einen weiten Temperaturbereich, eine verunreinigte und verschmutzte Atmosphäre sowie einen großen Bereich von Schocks und Vibrationen einschließt.

- Sicherheitstest
- Hauptfunktions-Wirkungsgrad
- Wartungsprogramme

Bild 14.10
Diagnose-Steuersystemanforderungen

Temperatur:	–50 °C bis +125 °C
Feuchtigkeit:	5% bis 100%
Mech. Schocks:	Größer als 30 g
Vibrationsbereiche:	Bis zu 5 g über einen Bereich von 5-30 Hz

Verschmutzung

Großer Betriebsspannungsbereich

Bild 14.11
Umgebungsbedingungen für Kfz-Anwendungen

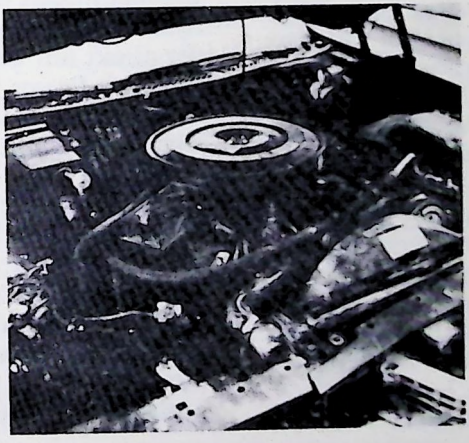


Bild 14.12
Verschmutzter Motorraum

Zentrales Mikroprozessorsystem
(Bild 14.13)

Unter Berücksichtigung der besprochenen Systemparameter und der Kfz-Umgebungsbedingungen wird nun untersucht, welcher spezifische Typ eines Mikroprozessors benötigt wird und wie sich dessen Eigenschaften mit den Automobil-Parametern vereinbaren lassen. Für den Einsatz eines Mikroprozessorsystems in einem Kraftfahrzeug gibt es zwei Grundkonzeptionen. Das erste, in Bild 14.13 dargestellte System arbeitet mit einem einzigen zentralen Mikroprozessor, der für die Steuerung der gesamten, vorher besprochenen Funktionsbereiche verantwortlich ist. Dieses System könnte so ausgeführt sein, daß die verschiedenen analogen Eingangssignale einem Multiplexer zugeführt würden, dessen Ausgang mit einem Analog-Digital-Umsetzer verbunden ist, der die nun digitalisierten Eingangs-Sensorsignale an den Datenbus des Mikroprozessors weitergeben würde. Darüber hinaus würde der Datenbus verschiedene Operationen durchführen, die vom Mikroprozessorsystem auf Veranlassung der Antriebselemente, die mit einem zentralen Datenbus verbunden sind, gewünscht werden. Obwohl diese Methode recht einfach ist und nur wenig Hardware erfordert, würde sie nicht allen angestrebten Systemzielen genügen. So besitzt das System z.B. keine Redundanz, wodurch man es nicht ohne weiteres mit genügender Ausfall-Sicherheit ausführen könnte. Da außerdem der Mikroprozessor 20 bis 30 Funktionen in einer bestimmten Zeit abzuwickeln hätte, wäre ein Modell mit sehr spezieller Architektur und hoher Ausführungsgeschwindigkeit sowie komplexer Ein-/Ausgabestruktur erforderlich.

Dezentrales Mikroprozessorsystem
(Bild 14.14)

Eine zweite Ausführungsmöglichkeit eines Mikroprozessorsystems ist in Bild 14.14 dargestellt. Hier werden anstelle eines einzelnen zentralen Mikroprozessors wie in Bild 14.13 mehrere, zweckgebundene Mikroprozessoren verwendet. Jeder dieser einzelnen Mikroprozessoren erfüllt die vorher besprochenen, unterschiedlichen Funktionen. So enthält dieses dezentrale Mikroprozessorsystem beispielsweise einen Mikroprozessor für die Motorsteuerung, einen für die Getriebesteuerung, einen weiteren für die Sicherheitseinrichtungen sowie einen für die Komfort-Funktionen. Weitere könnten für die Anpassung an zusätzliche Funktionen hinzugefügt werden.

Darüber hinaus befindet sich in diesem System ein "Haupt"-Mikroprozessor, der direkt mit dem Treiber-Interface verkehrt. Dieser Haupt-Mikroprozessor ist das höchste Ausführungsorgan des Diagnoseprogramms, der das System überwacht und den Fahrer auf jede Störung oder möglichen Fehler hinweist. Innerhalb dieses Systems ist es möglich, daß ein Mikroprozessor die Funktionen eines anderen mit übernimmt. Verglichen mit einem zentralen Mikroprozessorsystem bietet deshalb das dezentrale Mikroprozessorsystem mehr Flexibilität, um Systemausfälle zu verhindern.

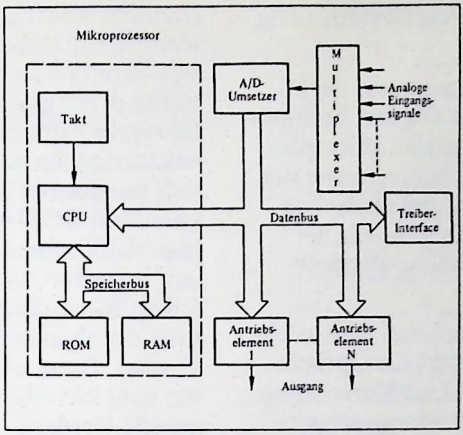


Bild 14.13
Zentrales Mikroprozessorsystem

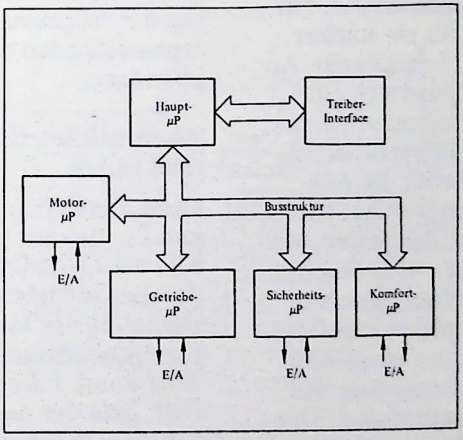


Bild 14.14
Dezentrales Mikroprozessorsystem

Mikroprozessor-Getriebesteuerung
(Bild 14.15)

Anhand der Untersuchung, was ein Mikroprozessor innerhalb des ihm bestimmten Bereichs des dezentralen Mikroprozessorsystems leisten könnte, lassen sich zahlreiche, wichtige Informationen gewinnen. Für diese Analyse wurde der Mikroprozessor für die Getriebesteuerung ausgewählt.

Vorab sei daran erinnert, daß eine solche Steuereinheit (Controller) die Optimierung der Drehmoment-Last-Kurve sowie die Begrenzung der Fahrzeuggeschwindigkeit und des Radschlupfes durchzuführen hat. Ferner könnte dieser Controller Teil eines Fahrzeug-Anpassungssystems sein, welches dem Fahrer erlaubt, in Abhängigkeit des unterschiedlichen Fahrverhaltens verschiedene Leistungs-Betriebsarten auszuwählen. Ein Ausführungsvorschlag für ein solches System ist in Bild 14.15 dargestellt. An der Eingangswelle befinden sich ein Drehzahl- und Drehmomentaufnehmer, außerdem an der zum Differential führenden Welle ein Sensor für die Ausgangsdrehzahl sowie ein weiterer Drehzahlaufnehmer an jeder der beiden Wellen für die Antriebsräder. Man beachte, daß die Drehzahl- und Drehmoment-Meßwerte einem Multiplexer zugeführt werden. Er schaltet sie auf einen Analog-Digital-Umsetzer, der sie über den Datenbus an die CPU weiterleitet. Das System enthält außerdem ein Stellglied für die Einstellung der Getriebeübersetzung, ein Stellglied für die Differential-Schlupfbegrenzung sowie eine zusätzliche Ein-/Ausgabeeinheit, die später erläutert wird.

Details einer Mikroprozessor-Getriebesteuerung
(Bild 14.16)

Um mehr Einblick zu erhalten und um das vorgeschlagene System definieren zu können, sind von dem zu untersuchenden Mikroprozessor in Bild 14.16 weitere Einzelheiten herausgezeichnet. Wie bereits besprochen, führen zum Getriebe drei Datenwege. Wenn eine auszuführende Aufgabe auf einer Aktion mit bekannten Zuständen beruht, kann es sich bei dem Steuerspeicher ganz einfach um einen ROM handeln. Der Adressierplan würde dann den notwendigen Ablauf aufrufen, während die Adressen selbst durch die Systemdaten oder eventuell durch externe Befehle modifiziert würden. Diese externen Befehle könnten über ein "Betriebsart"-Register angepaßt werden, das den Befehl in den Prozessor eingibt. Diese Eingänge könnten Statusbits, Unterbrechungen, Steuerleitungen oder auch lediglich reservierte Felder oder Datenleitungen sein.

Auf diese Weise wurde eine einfach programmierte Steuereinheit geschaffen, die ihr Programm entsprechend den Systemzuständen oder externen Befehlen abändert.

Zusätzliche Ein-/Ausgabeeinheit
(Bild 14.17)

Diese zusätzliche Ein-/Ausgabeeinheit kann in diesem System entsprechend Bild 14.17 realisiert werden. Sie verbindet die Getriebesteuereinheit mit dem Datenbus der Motor-, Sicherheits- oder Komfortsteuereinheit. In diesem Entwurf kann, falls eine Steuereinheit ausfällt, jede die Aufgaben der anderen übernehmen. Wenn eine solche Situation eintritt, schaltet der Selektor/Multiplexer den Datenbus der defekten Steuereinheit auf den Datenbus einer arbeitenden Steuereinheit. Dadurch läßt sich bei eingeschränktem Betrieb ein relativ sicherer Zustand auch für den Fall erreichen, daß vier der fünf Steuereinheiten ausfallen sollten.
Anhand der Blockschaltung des Systems und mit Hilfe einiger zusätzlicher Parameter kann nun der geeignetste Mikroprozessor für diese Anwendung ausgewählt werden.

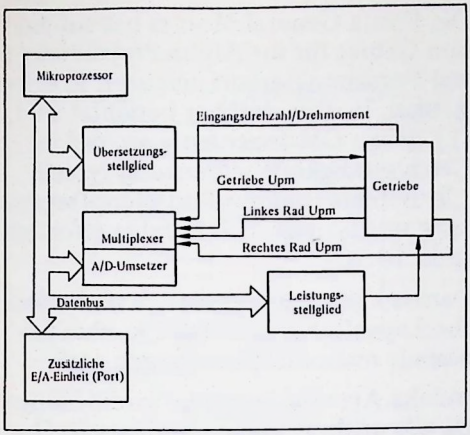


Bild 14.15
Mikroprozessor-Getriebesteuerung

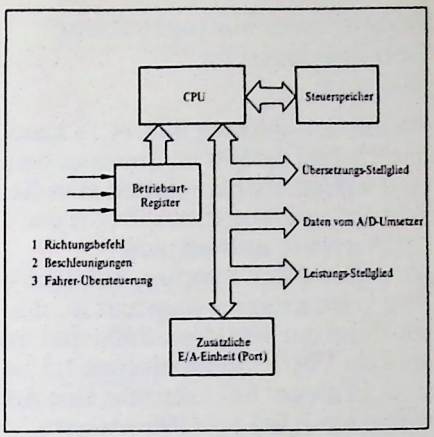


Bild 14.16
Details einer Mikroprozessor-Getriebesteuerung

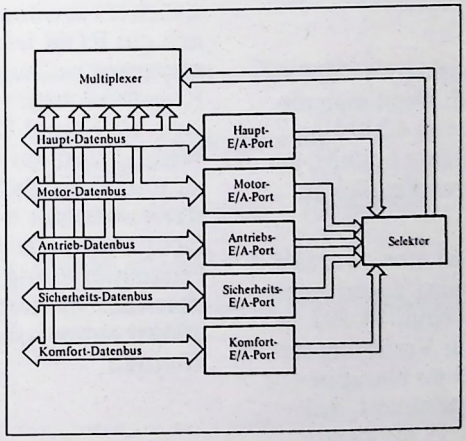


Bild 14.17
Zusätzliche Ein-/Ausgabeeinheit (E/A-Port)

Beispiel einer Mikroprozessor-Getriebesteuerung
(Bild 14.18)

Für das Getriebe nach Bild 14.18 kann nun die Zahl der Bits im Datenbus definiert werden. Hierzu ermittelt man die Zahl der Auflösungselemente, die der A/D-Umsetzer auflösen muß.
Wir wollen davon ausgehen, daß das System in ein Fahrzeug eingebaut ist, dessen Motor mit 4000 Upm dreht und dessen Fahr-Übersetzungsverhältnis 1:1 beträgt. In diesem Fall kann man eine Auflösung von 1 Teil aus 4096 erwarten. Dies erfordert umgesetzt eine Auflösung von 12 bit. Die anderen Fahrzeug-Parameter lassen sich in einer ähnlichen Weise umsetzen.

Motor-Steuereinheit-Geschwindigkeit
(Bild 14.19)

Was ist zur Geschwindigkeit der Motorsteuereinheit zu sagen? Wenn man die vorgenannte Drehzahl von 4000 Upm auf die Motorsteuereinheit bezieht, gewinnt man einen tieferen Einblick in dieses Problem.

Es sei angenommen, daß man zwischen zwei Kurbelwellen-Umdrehungen einen Zustand zu ändern hat (Bild 14.19). In diesem Fall wird die zur Verfügung stehende Ansprechzeit 15 ms betragen. Wenn man nun weiter annimmt, daß eine Entscheidung, die 15 ms dauern darf, etwa 100 Mikroprozessor-Befehlszyklen beanspruchen wird, läßt sich folgern, daß ein relativ langsamer Prozessor benötigt wird. Tatsächlich muß ein solcher Prozessor eine Speicher-Zykluszeit von etwa 150 μs oder weniger aufweisen.

Mikroprozessor-Speicherkapazität
(Bild 14.20)

Es verbleibt nun noch ein wichtiger Parameter, nämlich die Speicherkapazität.

Die Firma General Motors hat auf diesem Gebiet für ihr Alpha-Projekt sehr viel Forschungsarbeit geleistet. In einem Artikel, in dem darüber berichtet wird (1), geben GM-Ingenieure an, daß die Geschwindigkeitsbegrenzung typisch 120 Befehle und die Radschlupfbegrenzung nur typisch 100 Befehle erfordern (Bild 14.20).

Daraus kann man ersehen, daß die Speicherkapazität etwa 1 bis 2 k, aber keinesfalls mehr als 4 k betragen muß.

Welche Art von Speicher wird benötigt? Da alle arithmetische und logische Operationen mit augenblicklichen Daten durchgeführt werden, um die neue Subroutine bzw. Steueradresse zu erhalten, ist kein RAM-Speicher zur Datenspeicherung erforderlich. Darüber hinaus ist auch keine Modifizierung des Steuerspeichers erwünscht. Deshalb wird lediglich ein ROM benötigt, falls ein Mikroprozessor mit angemessenen internen Registern ausgesucht wird, die als Stack bzw. als Stapel-Register ausgeführt sind. Wenn davon ein Register als Befehlszähler und eines als Stack- oder Indexregister verwendet werden können, vereinfacht sich die Implementierung und Programmierung erheblich. Noch einfacher wäre es, wenn ausgewählte Register inkrementiert und dekrementiert werden könnten.

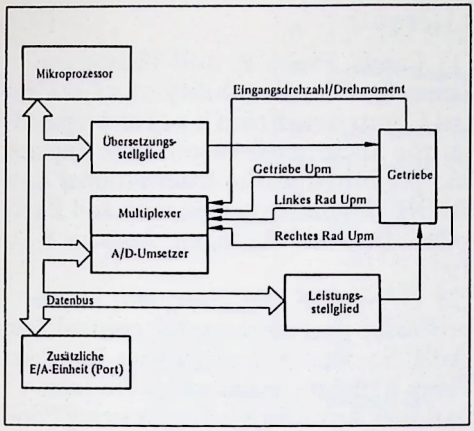


Bild 14.18
Beispiel einer Mikroprozessor-Getriebesteuerung

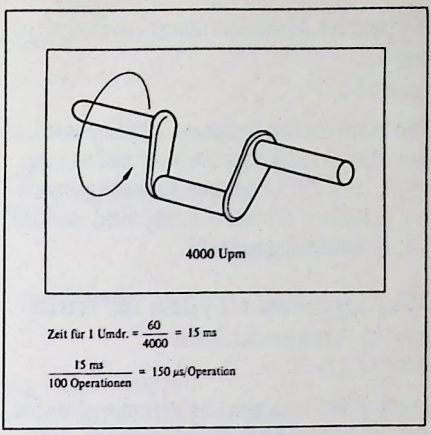


Bild 14.19
Geschwindigkeit der Motor-Steuereinheit

Speicherkapazität – Gleich oder weniger als 4 k Bits

Geschwindigkeitsbegrenzung – 120 Befehle

Radschlußfbegrenzung – 100 Befehle

Register als internes RAM

Keine Online-Programmierung = Festwert-speicher (ROM)

Bild 14.20
Mikroprozessor-Speicherkapazität

Typische Mikroprozessor-Parameter
(Bild 14.21)

Die wichtigsten technischen Eigenschaften, die ein Mikroprozessor aufweisen muß, und die Umgebungsbedingungen bei denen er arbeiten muß, sind in Bild 14.21 zusammengefaßt.

Mikroprozessor-Typen für Automobil-Anwendungen
(Bild 14.22)

Diese Faktoren sind bestimmend dafür, daß in Automobil-Anwendungen 12, oder 16 Bit Mikroprozessoren, oder auch 4-Bit Mikroprozessorelemente eingesetzt werden sollten. Einige mögliche Typen zeigt Bild 14.22.

Die für diese Anwendung nötige Software ist in mancher Hinsicht einzigartig. Der Programmieraufwand für voneinander abhängiger Moduln, die sich in ihren Steuer-Anforderungen überlappen, ist recht umfangreich. Um die Speicher-Anforderungen herabzusetzen, sollte die Programmierung in einer Assembler- oder Maschinensprache durchgeführt werden. Darüber hinaus wird ein hervorragender Simulator benötigt. Für diesen erheblichen Programmieraufwand ist es sinnvoll, für das Softwaresystem einen Hardware-Simulator zu bauen. Dadurch ist es möglich, die Software unter möglichst natürlichen Umgebungsbedingungen auszutesten.

Zusammenfassung

Dieses Kapitel hat gezeigt, daß der Einsatz von Mikroprozessoren in Automobil-Anwendungen einige einzigartige und herausfordernde Entwurfskonzepte verlangt. Die hierbei gewonnenen Kenntnisse können aber auch zugleich sehr nützlich bei anderen Anwendungen sein, die relativ anspruchsvolle Steuereinheiten benötigen.

Literatur

(1) Caiati, Frank P. und Thompson, James F.: The Feasibility of a Car Central Computer. From a paper prepared for the Society of Automotive Engineers and presented at the International Automotive Engineering Congress and Exposition, Detroit, Michigan, January 8-12, 1973.

(2) G. NOVIS and J. Bryant: Mikroprozessor performs engine control functions. Society of Automotive Engineers. From a paper presented at the International Automotive Engineering Congress and Exposition, Cobo Hall, Detroit, Michigan, February 28 – March 4, 1977

- 12- oder 16-Bit-Architektur
- Ausführungszeiten von gleich oder weniger als 150 μs
- Stapelregister
- Vielseitige Adressierung
- Eine Betriebsspannung
- Umweltbedingungen
  −50 bis +125 °C
  5 bis 100 % relative Feuchtigkeit
  30 g Schock
  5 g Vibration
  Starker Verschmutzung und Korrosion ausgesetzt

Bild 14.21
Typische Mikroprozessor-Parameter

- SBP9900 oder 0400A
- 2901
- 9980
- 1801

Bild 14.22
Mikroprozessor-Typen für Automobil-Anwendung

KAPITEL 15

MIKROPROZESSOREN IN DER UNTERHALTUNGS-INDUSTRIE

Dieses Kapitel beschäftigt sich damit, wie Mikroprozessoren den Markt für Unterhaltungsgeräte und Spielzeuge beeinflussen können. Es zeigt auf, wie sich die Fähigkeiten des Mikroprozessors auf die Spieltheorie auswirken und wie sich die Kosten bei typischen Anwendungen verhalten. Als spezifisches Beispiel dient ein Schach-Kursus.

Mikroprozessoren in der Unterhaltungsindustrie

In diesem Kapitel wird auf den Anwendungsbereich von Unterhaltungsgeräten eingegangen. Bezogen auf den Umsatz dürfte dies eines der interessantesten Anwendungsgebiete sein. Unterhaltungssysteme weisen Anforderungen auf, die sich von denen anderer Anwendungen unterscheiden.

Voraussetzungen für Unterhaltungsgeräte auf Mikroprozessor-Basis
(Bild 15.1)

In Bild 15.1 sind die verschiedenen Voraussetzungen für Unterhaltungsgeräte auf Mikroprozessor-Basis zusammengefaßt. Die über alles dominierende Voraussetzung sind niedrige Kosten. Die Kosten müssen niedrig sein, und zwar etwa 1500 DM oder weniger für Unterhaltungsgeräte, die für einzelne Verbraucher gedacht sind sowie etwa 1500 DM bis 15 000 DM für Institute und Gesellschaften. Die Geräte müssen außerdem sehr handlich und praktisch sowie extrem einfach zu bedienen sein. Darüber hinaus sollte die Anlage sowohl aus der Sicht des Marketing wie auch des Benützers erweiterbar sein.

Unterhaltungs-Anwendungen
(Bild 15.2)

Im Bereich der Unterhaltung gibt es zahlreiche Aufgaben, die von Systemen auf Mikroprozessor-Basis erfüllt werden können. Diese Aufgaben lassen sich generell in zwei Kategorien aufteilen, und zwar in Spiele sowie in Schulungs- bzw. Ausbildungs-Anwendungen (Bild 15.2).

Unterhaltungsspiele selbst können unterteilt werden in die allgemeinen Bereiche der Geschicklichkeitsspiele, Glücksspiele und Strategiespiele. Die meisten Geschicklichkeitsspiele beruhen auf ballistischen oder Flugbahn-Gleichungen. Bei dieser Art von Spielen besteht die Hauptaufgabe des Mikroprozessors darin, die Gleichungen zu lösen und die Anzeigeeinheit (Display) zu steuern.

Glücksspiel
(Bild 15.3)

Die zweite Kategorie der Spiele hat mit dem Glück zu tun. Bei diesen Anwendungen muß der Mikroprozessor gewöhnlich eine Zufallszahl erzeugen und auf geeignete Ereignisfolgen für jede erzeugte Zahl springen; ferner muß er das Display oder andere Ausgabeeinheiten steuern.

- DM 1500,– oder weniger für Privat personen
- DM 1500,– bis DM 15.000,– für Gesellschaften
- Interessante Ein-/Ausgabe-Eigenschaften
- Minimaler Zeitabstand, Lärm usw.
- Gerechtfertigte Erst-Anschaffung
- Anpaßfähig an verschiedene Bedürfnisse
- Tragbare Zusatzkosten

Bild 15.1
Voraussetzungen für Unterhaltungsgeräte auf Mikroprozessor-Basis

- Spiele
  - Geschicklichkeit
  - Glück
  - Strategie
- Ausbildung

Bild 15.2
Unterhaltungs-Anwendungen

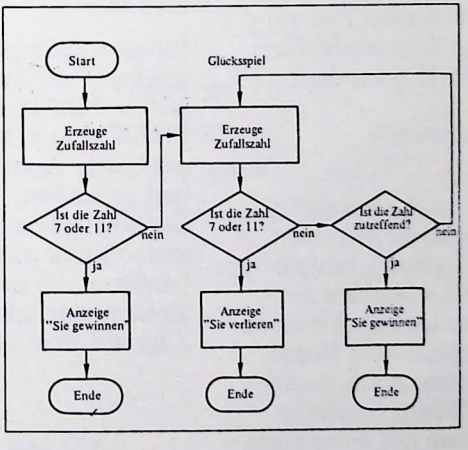


Bild 15.3
Glücksspiel

Strategiespiel
(Bild 15.4)

Die Strategiespiele als dritte Kategorie schließen Fußball, Tic-Tac-Toe, Dame und Schach ein. Die Durchführung bzw. Implementierung solcher Spiele ist besonders schwierig, da einerseits im Mikroprozessorsystem eine nahezu unbegrenzte Programmierung und andererseits eine gut funktionierende interaktive Ein-/Ausgabeeinheit vorhanden sein müssen. Üblicherweise bleibt jedes digitale System weit hinter den Erwartungen zurück, wenn man es auch an den dümmsten Spieler anpassen muß. Das trifft besonders auf die Spiele der dritten Kategorie zu. Dennoch können gewisse Spiel-Einschränkungen vorgenommen werden. Als Beispiel eines Systementwurfs wird in diesem Kapitel ein System erörtert, anhand dessen ein Benützer die Regeln eines Spieles erlernen kann, wobei es sich in diesem Fall um einen hypothetischen Schachspiel-Kursus entsprechend Bild 15.4 handelt.

Schachspiel-Einzelheiten
(Bild 15.5)

Ein Schachspiel-Lehrgang kann mehr oder weniger anspruchsvoll aufgebaut sein. Wie in Bild 15.5 gezeigt, besteht die einfachste und gut erlernbare Art darin, daß das System die verschiedenen Schachzüge einschließlich ihrer Bedeutung erklärt. Das System würde in diesem Fall Schritt für Schritt durch die Züge führen, zusammen mit geeignetem, beschreibendem Text. Das Lehren von Eröffnungszügen würde sich in ähnlicher Weise abspielen, allerdings wäre hierzu ein wesentlich umfangreicheres Programm und somit auch mehr Speicherkapazität erforderlich, als es das Grundsystem aufweist. Darüber hinaus sind auch Programme zur Demonstration grundlegender Strategien möglich, die wiederum – bei gleichbleibendem Hardwareaufwand – einen höheren Programmieraufwand und damit auch einen größeren Programmspeicher erfordern.
Über die Durchführbarkeit von Strategiespielen entscheidet nicht der Mikrocomputer oder ein eventuell erheblich größeres digitales System, sondern das Problem liegt in dem enorm hohen Programmier- und Speicheraufwand, wenn man alle möglichen Züge und Gegenzüge mit einbeziehen will. Beim Schachspiel beispielsweise hat jeder Spieler für den ersten Zug 20 unterschiedliche Setzmöglichkeiten, so daß die ersten Züge insgesamt 400 verschiedene Kombinationen ergeben. Auch ein sehr kurzes Spiel erfordert bereits eine astronomische Zahl von Kombinationen, die alle gespeichert und folgerichtig ausgeführt werden müssen.

Eingabeeinheiten
(Bild 15.6)

Einige mögliche Eingabeeinheiten, die die Basis zur Systemdefinition bilden, sind in Bild 15.6 angegeben. Die Tastatur hat durch Telefon und Taschenrechner breite Anerkennung und Bekanntheit gefunden. Obwohl die Tastatur-Codierung von Schachzügen auf den ersten Blick ungeschickt erscheinen mag, benützen die meisten Schach-Lehrbücher alphanumerische Schreibweise, um die Züge auf dem Brett zu kennzeichnen.

Bild 15.4
Strategiespiel

- Züge erklären
- Eröffnungszüge erklären
- Grundlegende Strategie demonstrieren

Bild 15.5
Schachspiel-Einzelheiten

- Brett mit Sensoren
- Tastatur
- Spielfiguren
- Drehknöpfe/Schalter
- Hebel

Bild 15.6
Eingabeeinheiten

Tastatur-Schachschreibweise
(Bild 15.7)

Mit der Tastatur als System-Eingabeeinheit können die Züge entsprechend dem Format in Bild 15.7 codiert bzw. verschlüsselt werden. In der Schachsprache bedeutet das, "Bauer zum König's Läufer vier" bzw. deutlicher ausgedrückt: Ziehe mit dem Bauern vor des König's Läufer zum vierten Quadrat des Läufers.

Um mit dieser Schreibweise im System etwas anfangen zu können, wird mit dem linken Buchstaben (in diesem Fall B für Bauer) eines der sechs Schachfigur-Symbole bezeichnet; sie können jeweils durch ein 4-Bit-Wort dargestellt werden. Die Bezeichnung KL 4 kann entweder zur Kennzeichnung einer Reihe oder Spalte herangezogen werden, wofür jeweils eine 3-Bit-Adresse genügt, oder aber sie dient als absolute Brettquadrat-Indexzahl. Die zweite Methode erfordert 64 Kennzeichen, was 6 Bit entsprechen würde. Mit der Reihen- und Spalten-Methode würde sich die Programmierung vereinfachen und es könnte jeweils ein Standard-4-Bit-Datenwort verwendet werden. Dadurch kann jeder Zug auf dem Spielbrett kurz und knapp codiert und über eine alphanumerische Standard-Tastatur eingegeben werden.

Schachspiel-Ausgabeeinheit
(Bild 15.8)

Die Ausgabeeinheit für solche Spiele steht bereits zur Verfügung, denn nahezu alle Benützer besitzen ein Fernsehgerät (Bild 15.8a). Hier hat das System lediglich die Aufgabe, die zur Ansteuerung des Displays geeigneten Signale zu erzeugen. Diese Interface-Steuerung kann vom Mikroprozessor übernommen werden, wobei sich dessen Ausgang direkt mit dem Antenneneingang des Fernsehempfängers (Bild 15.8b) verbinden läßt. Aus diesem Grunde ist die Installation einer neuen Unterhaltungsanlage in sehr kurzer Zeit und ohne Schwierigkeiten möglich.

Schachspiel-Blockschaltung
(Bild 15.9)

In Bild 15.9 sind die wichtigsten vier Baugruppen für ein solches Lehrsystem dargestellt. Im Speicher sind die Symbole, die Programme, Zug-erklärende Algorithmen sowie eventuell Videotexte gespeichert. Der Mikroprozessor hat beispielsweise für den Programmablauf sowie für die Steuerung des Speichers und des Displays zu sorgen. Um die Funktionen des Mikroprozessors und des Speichers vollständig festzulegen, muß noch das Betriebssystem definiert werden.

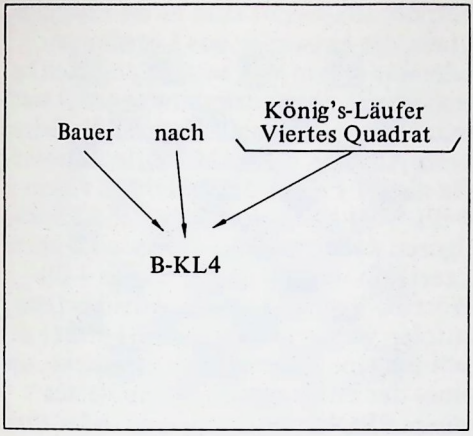


Bild 15.7
Tastatur-Schachschreibweise

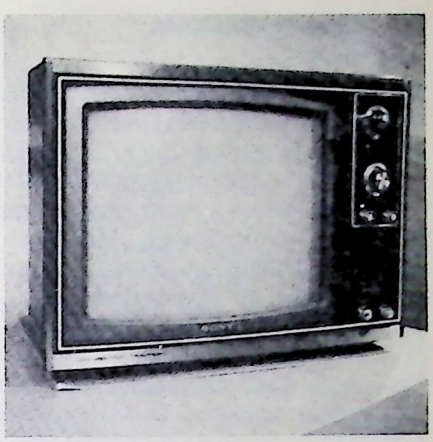


Bild 15.8a
Schachspiel-Ausgabeeinheit

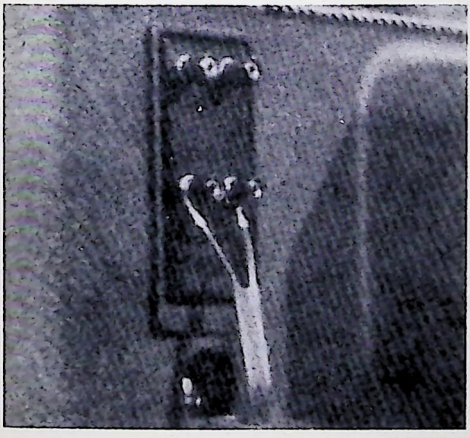


Bild 15.8b
Schachspiel-Ausgabeeinheit

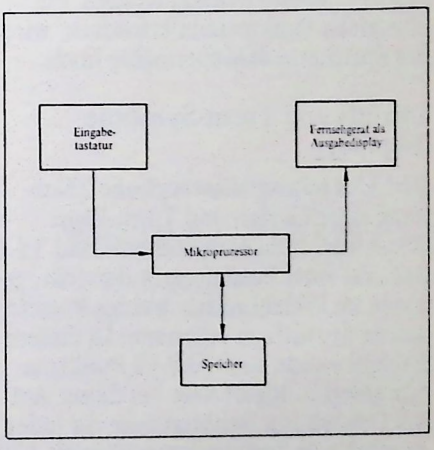


Bild 15.9
Schachspiel-Blockschaltung (Lehrsystem)

Schachbrett-Quadrate
(Bild 15.10)

Für ein Schach-Lehrsystem wird zur Darstellung des Spielbrettes ein Gitter von 64 Quadraten benötigt (Bild 15.10). Da es sich nicht verändert, könnte es in der TV-Steuereinheit fest verdrahtet werden. Um sich jedoch ein gewisses Maß an Flexibilität zu erhalten, wird dieses Muster zusammen mit den anderen erforderlichen Symbolen im Speicher des Systems gespeichert.
Die Schachfigur-Symbole sind also ebenfalls im Systemspeicher enthalten. Wenn sehr einfache Symbole verwendet werden, können die Anforderungen an den Systemspeicher niedrig gehalten werden. In diesem Fall werden jedoch darunter die Attraktivität und die Wirkung der Anzeige leiden. Wenn allerdings reich verzierte Muster benützt werden, die eine große Punktmatrix erfordern, wird der Speicherbedarf übermäßig hoch.

Läufer- und Turm-Symbole
(Bild 15.11)

Bild 15.11a zeigt eine typische Zeichnung einer Läufer- und Turm-Figur. Wenn man diese entsprechend Bild 15.11b in Form einer Punktmatrix darstellt, genügen im Prinzip relativ wenige Punkte, um sie deutlich zu erkennen. In diesem Beispiel wurde eine 10 x 12-Punktmatrix gewählt. Bleibt man bei dieser Art der Darstellung, benötigt man für jedes Symbol eine Speicherkapazität von 120 Bits. Glücklicherweise gibt es beim Schachspiel nur sechs unterschiedliche Spielfiguren, so daß sich die Speicher-Anforderungen in Grenzen halten.

Displaybetrieb-Anforderungen
(Bild 15.12)

Sowohl das Schachbrett- als auch die Schachfigur-Muster (Symbole) werden alle für das Display benötigt. Daraus ergeben sich die in Bild 15.12 definierten Speicher-Anforderungen für den Displaybetrieb. Insgesamt sind es 64 Brettquadrate, die entweder aus Leerräumen oder aus einem von sechs Symbolen bestehen, die entweder schwarz oder weiß sein können. Deshalb sind für die Schachbrett-Anzeige 64 RAM-Worte notwendig, die die 64 Felder darstellen. In einem 4-Bit-Wort kann man jede der 12 Spielfiguren zusätzlich der Adresse für einen Leerraum unterbringen, da ein 4-Bit-Wort 16 Kombinationen erlaubt. Demzufolge würde jedes dieser 64 Worte zu je 4 Bit eine Adresse repräsentieren, die eines der Punktmatrix-Symbole aus einem PROM abrufen würde; oder aber die Adresse würde bewirken, daß ein bestimmtes Spielfeld nicht besetzt wird bzw. frei bleibt.

Displayspeicher-Anforderungen
(Bild 15.13)

In Bild 15.13 sind die Speicheranforderungen, bezogen auf die Anzeige, zusammengefaßt. Für das Spielfeld werden 64 x 4 oder 256 Bit RAM und für die sechs Schachfigur-Symbole 6 x 120 oder 720 Bit ROM benötigt. Um eine volle Anzeige zu erhalten, muß die Display-Steuerung das 64-Worte-RAM auf einer Zeichen-pro-Zeichen-Basis und in einer Zeile-pro-Zeit-Abhängigkeit durchlaufen. Dadurch werden dem System einige Geschwindigkeits-Anforderungen aufgebürdet.

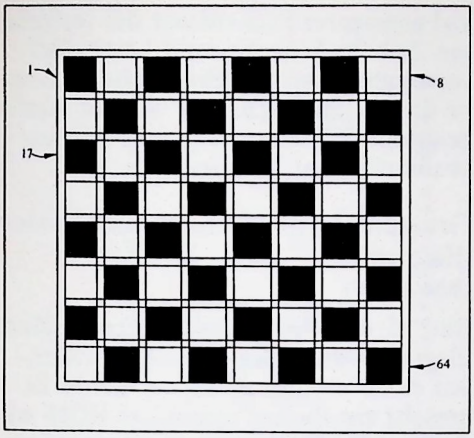


Bild 15.10
Schachbrett-Quadrate

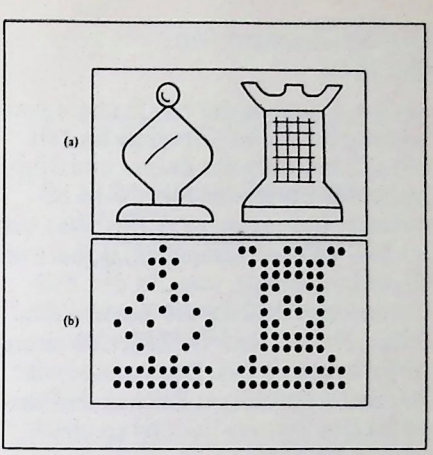


Bild 15.11
Läufer- und Turm-Symbole

- 64 Brettquadrate
- Figursymbole
- 6 Weiße
- 6 Schwarze
- 1 Leerraum

Bild 15.12
Displaybetrieb-Anforderungen

- 64 x 4 oder 256 Bits RAM
- 6 x 120 oder 720 Bits ROM

Bild 15.13
Displayspeicher-Anforderungen

Display-Zeichenform
(Bild 15.14)

Die zur Abtastung der 64 Quadrate eines Fernsehbildschirms erforderliche Zeit läßt sich mit Hilfe der Zeilen- und Spaltenform entsprechend Bild 15.14 bestimmen. Das gesamte Zeichen setzt sich aus 16 TV-Zeilen zusammen, wobei zwei für die Umrandung, zwei für den Zwischenraum und 12 für die Zeichenhöhe dienen. Die normale TV-Zeilenabtastung erfolgt in etwa 64 µs. Demzufolge wird für einen vollständigen Zeichen-Aufbauzyklus eine Zeit von 16 x 64 µs gleich 1024 µs benötigt.
Da acht solcher 16 x 16 Bildpunkte quer zum Bildschirm liegen, ist ein Register notwendig, in dem eine ganze Zeichenreihe zwischengespeichert wird. Hierfür werden 16 Zeilen nach unten und 16 x 8 oder 128 Spalten benötigt. Wenn dieser Bild-Aufbau mit zwei 16 x 128-Bit-Registern durchgeführt wird, kann man das eine laden,während der Inhalt des anderen auf dem Bildschirm angezeigt wird.

Display-Zykluszeit
(Bild 15.15)

Unter Verwendung eines typischen 4-µs-Speicherzyklus für den Mikroprozessor sind in Bild 15.15 die erforderlichen Speicherzyklen angegeben. Da es pro Zeichen sechs Leerspalten gibt, sind zehn Zyklen notwendig, um die zehn Spielfeld-Punktspalten aufzurufen, die von Interesse sind. Und mit einer Zeichenhöhe von 12 Punktreihen werden 10 x 12 oder 120 Zyklen benötigt, um die Zeicheninformation zu erzeugen. Daraus ergibt sich, daß 128 Speicherzyklen zu je 4 µs oder 512 µs benötigt werden.
Verglichen mit der Gesamtzeit von 1024 µs beansprucht das Display etwa 50 % der verfügbaren Speicherzeit, um die Daten für das Display auszulesen. Da der Spieler von Hand eingibt, wird ein viel geringerer Prozentsatz der verfügbaren Zeit für Eingabe- und Lesezyklen verwendet. Somit steht, nachdem Eingabe und Ausgabe-Display bedient wurden, genügend Speicherzeit zur Programmausführung zur Verfügung.

Grundlegendes Schach-Lehrsystemdiagramm
(Bild 15.16)

Bild 15.16 zeigt das grundlegende Blockdiagramm eines Schach-Lehrsystems, wie es bis jetzt besprochen wurde. Es besteht ein Bedarf sowohl an ROM- wie auch an RAM-Speichern mit einer Speicher-Zykluszeit von 4 µs. Mit einem längeren Speicherzyklus würde die Bedienung des Displays Schwierigkeiten bereiten.

Speicher-Organisation
(Bild 15.17)

Eine weitere Voraussetzung ist das Vorhandensein eines Programmspeichers. Für die Programmspeicherung benötigt das System einen möglichst großen Speicher. Da man hierdurch wesentlich mehr Strategie-Variationen behandeln könnte, würde dies zu einer erheblichen Bereicherung des Spiels führen. Um jedoch die grundlegenden Züge und Regeln zu erklären, ist nur eine kleine Zahl fester Algorithmen und Konstanten notwendig. Daraus läßt sich folgern, daß auch der Programmspeicher – wie Bild 15.17 zeigt – aus Festwertspeichern (ROM's) und Schreib-/Lesespeichern (RAM's) besteht. Das ROM würde einerseits die grundlegenden Regeln und Züge enthalten, die sich nicht ändern, und das Urlader (Bootstrap-)Programm. Im RAM dagegen wären die Strategie-Programme gespeichert, die von externen Datenquellen (Kassette ect.) geladen werden.

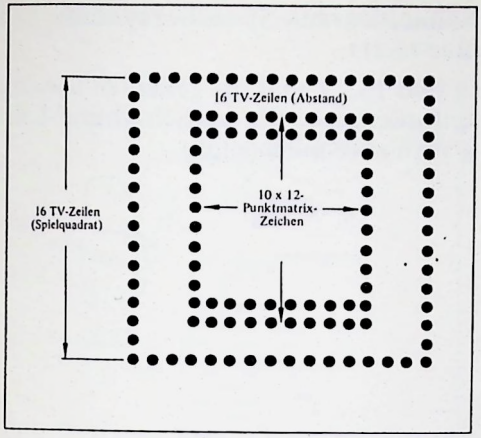


Bild 15.14
Display-Zeichenform

- 8 Speicherzyklen für die Zeichendefinition
- 96 Speicherzyklen für die Zeichenerzeugung
  (8 Zwischenräume x 12 Punkt-Kolonnen)
- 104 Gesamt-Speicherzyklen pro Zeichenbild

Bild 15.15
Display-Speicherzyklen

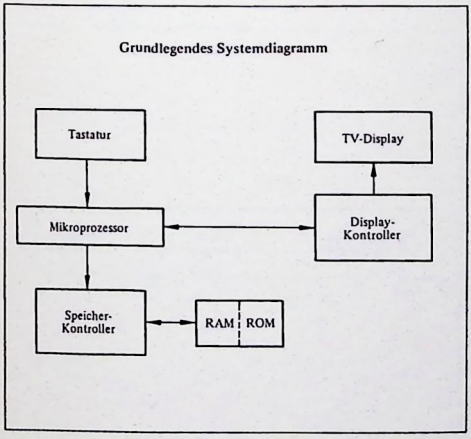


Bild 15.16
Grundlegendes Schach-Lehrsystemdiagramm

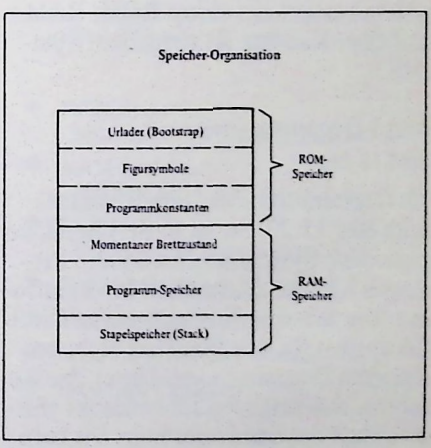


Bild 15.17
Speicher-Organisation

Beispiele für die Programmeingabe
(Bild 15.18)

Zur Eingabe des Programms in das System kann beispielsweise eine gelochte Karte in Verbindung mit einem preisgünstigen sog. "Durchfall"-Kartenleser oder vielleicht auch einem handelsüblichen Kassettenrecorder verwendet werden (Bild 15.18). Eine Kassette kann als großer Sekundärspeicher dienen, außerdem wäre es möglich, daß sie akustische Hinweise und Instruktionen enthält.

Grundlegendes Systemdiagramm mit Kassette
(Bild 15.19)

Ein Unterhaltungssytem, das von einem Kassettenspeicher Gebrauch macht, ist in Bild 15.19 dargestellt. Das Speichersystem besteht aus einem ROM, RAM und einer Kassette als virtuellem Speicher.

Band-Organisation
(Bild 15.20)

Die Organisation einer Bandkassette zeigt Bild 15.20. Sie ist in drei Bereiche gegliedert: Systeminformationen, Programm und die akustischen Informationen. Die Systeminformationen bereitet das System für den Empfang eines besonderen Programms vor. Unter der Annahme, daß keine Fehler entdeckt wurden, läuft das Band vom Start bis zum Ende des Digital-Programms durch. Falls jedoch Fehler aufgetreten wären, würde es zurücklaufen, danach von neuem beginnen oder der Bedienperson das Signal "Fehler" mitteilen. Wenn schließlich die Programmdaten mit Erfolg eingelesen worden sind, schaltet das Gerät vom Lade- in den Spielbetrieb um. Während dieser Operation könnten – gesteuert vom Mikroprozessor – dem Benützer verbale Hinweise und Instruktionen akustisch übermittelt werden.

Schachkursus-Speichersystem
(Bild 15.21)

In Bild 15.21 sind die gesamten Speicher-Anforderungen für das Schachspiel-Lehrsystem zusammengefaßt.

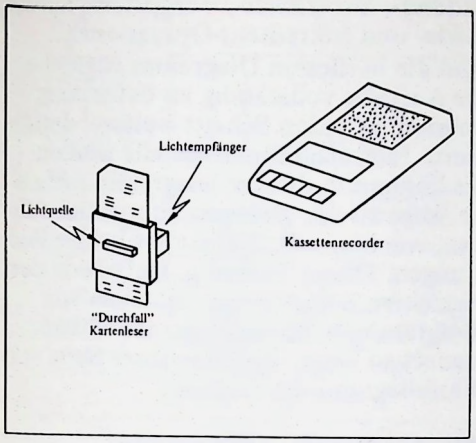


Bild 15.18
Beispiele für die Programmeingabe

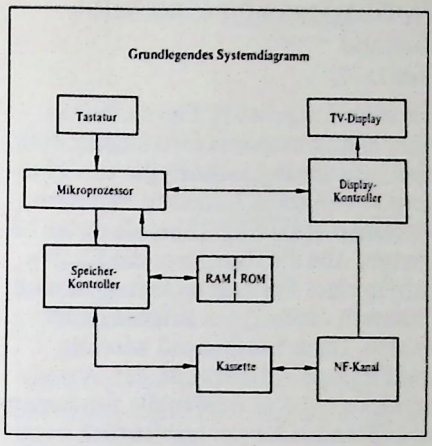


Bild 15.19
Grundlegendes Systemdiagramm mit Kassette

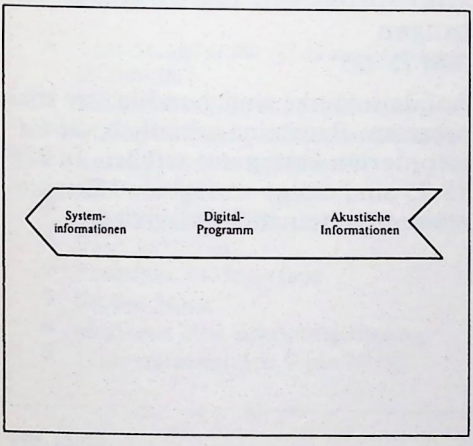


Bild 15.20
Band-Organisation

- ROM-Speicher
  Minimal 1024 Worte zu je 8 Bit
- RAM-Speicher
  Minimal 4096 Worte zu je 8 Bit
- Bandkassette
  Unbegrenzte Speicherkapazität
- Zykluszeit kleiner oder gleich 4 μs
- DMA erwünscht
- Mehrere Adressierungsarten
- Interrupt-Fähigkeit

Bild 15.21
Schachkursus-Speichersystem

Flußdiagramm für Einschalt-Zustand
(Bild 15.22)

Um die erforderlichen Eigenschaften eines Mikroprozessors festzulegen, definiert man anhand einer Reihe von Flußdiagrammen seine Aufgaben. Mit dem Diagramm in Bild 15.22 sei als erstes begonnen. Alle die Routinen, die für den Betrieb eines Systems unabdingbar sind, müssen in dieser Form aufgezeichnet werden. Darauf aufbauend wird die grundlegende Arbeitsweise des Systems entwickelt und es werden die fundamentalen Entscheidungen hinsichtlich benötigter Logik, arithmetischer Funktionen usw. getroffen. Ferner können zu diesem Zeitpunkt die Softwaremoduln aufgeteilt und der Programmieraufwand abgeschätzt werden.

Beispiel eines Schachzug-Flußdiagramms
(Bild 15.23)

Das in Bild 15.23 dargestellte Flußdiagramm dient dazu, den von einem Studenten ausgeführten Schachzug auszuwerten. Um die Frage "Ist der Zug legal? " zu beantworten, muß ein Unterprogramm entwickelt werden, das den Mikroprozessor die folgenden Aufgaben ausführen läßt: Zuerst muß der Mikroprozessor die Schachfigur identifizieren, die aufgrund einer 4-Bit-Tastatureingabe erstmals gesetzt wurde. Danach holt der Mikroprozessor alle legalen Züge dieser Figur aus dem Speicher und prüft durch einfache Addition oder Subtraktion der Schachfelder-Indexzahlen, ob diese Änderung gültig war. Schließlich überprüft der Mikroprozessor die dazwischen liegenden Felder bzw. Quadrate daraufhin, ob eine andere Figur den beabsichtigten Zug blockiert. Diese Aufgaben gehen weder über die Forderung nach einem 4-Bit-Wort hinaus, noch verlangen sie vom Mikroprozessor mehr als einfache Addier-, Subtrahier-, Vergleichs-, Sprung-, Lade- und Inkrement-Operationen.
Um die in diesem Diagramm angegebene Aufgabe vollständig zu definieren, müssen für jeden Schritt weitere, detaillierte Flußdiagramme erstellt werden. Es können in diesem Diagramm mehrere Wege zu der gleichen Subroutine führen, wie hier z.B. die beiden Fehler-Meldungen. Dieser Vorgang, Definition der einzelnen Schritte und Erstellen von Diagrammen auf niedrigerem Niveau, dauert so lange, bis eine ganze Serie von Flußdiagrammen vorliegt.

Mikroprozessor-Parameter
(Bild 15.24)

In Bild 15.24 sind die Parameter für das Schachspiel-Lehrsystem zusammengestellt.

Mikroprozessor-Hardwarebetrachtungen
(Bild 15.25)

Auf dem Markt sind verschiedene Mikroprozessor-Bausteine erhältlich, die die geforderten Parameter erfüllen. In Bild 15.25 sind einige verfügbare Mikroprozessoren miteinander verglichen.

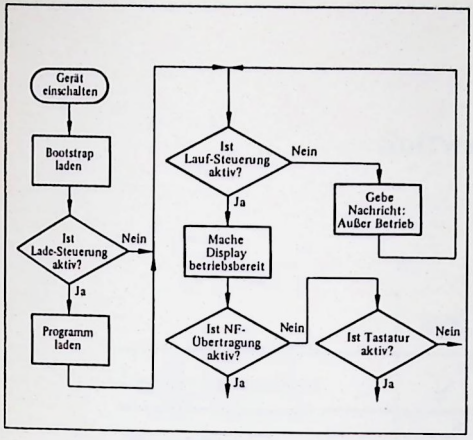


Bild 15.22
Flußdiagramm für "Einschalt"-Zustand

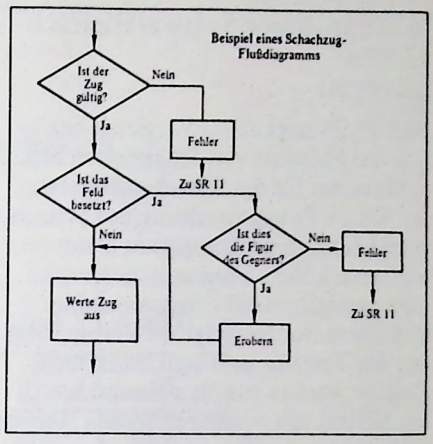


Bild 15.23
Beispiel eines Schachzug-Flußdiagramms

- 4-Bit-Struktur für ALU und Daten (Minimum)
- 8K Speicherkapazität (Minimum)
- 64K Speicherkapazität (erwünscht)
- Kurze Speicher-Zykluszeit (etwa 4 μs)
- Interrupt-Fähigkeit
- DMA bevorzugt
- Einfaches TV-Interface
- Großer Stack
- Möglichst eine Betriebsspannung
- Temperaturbereich 0 bis 70 °C

Bild 15.24
Mikroprozessor-Parameter

Hardware-Vergleich

	8080	6800	9900
Wortlänge	8	8	1, 8, 16
Speicher	64k	64k	64k
Takt	2 MHz	1 MHz	3 MHz
Interrupts	8	2	17
DMA	✓	✓	✓
E/A-Bausteine	✓	✓	✓
Betriebsspannungen	3	1	3
Temperaturbereich	0 - 70 °C	0 - 70 °C	0 - 70 °C

Bild 15.25
Mikroprozessor-Hardwarebetrachtungen

Mikroprozessor-Softwarebetrachtungen
(Bild 15.26)

Bild 15.26 zeigt einen Vergleich bezüglich der Software von erhältlichen Mikroprozessoren für die Schachspiel-Parameter. Da die Programmierung des Schachspiel-Lehrsystems komplexer Natur ist, wird eine höhere Programmiersprache oder wenigstens ein Cross-Assembler und Simulator benötigt. In Verbindung mit der Tastatur und dem Bildschirm-Display wäre es zudem zu empfehlen, ein System mit einem residenten Assembler und Monitor auszuwählen.

Auf dem Gebiet der Software gibt es einen eindeutigen Spitzenreiter; dennoch verfügen immerhin alle Mikroprozessoren über einen Cross-Assembler und Simulator. Somit ist jeder dieser Mikroprozessoren in der Lage, die Software- und Hardware-Anforderungen zu erfüllen.

Zusammenfassung

Dieses Kapitel untersuchte den Markt für Unterhaltungsspiele und spezifizierte die Anforderungen. Danach wurde als anspruchsvolles Beispiel ein Schachspiel-Lehrsystem entwickelt. Hierfür wurden über Blockschaltungen die Mikroprozessor-Anforderungen herausgearbeitet. Schließlich wurden einige auf dem Markt erhältliche Mikroprozessor-Typen bezüglich Hardware und Software untersucht, um die Auswahl des für diese Aufgabe geeigneten Mikroprozessors zu erleichtern.

Software-Vergleich

	8080	6800	9900
Cross-Assembler	✓	✓	✓
Simulator	✓	✓	✓
Höhere Programmiersprache	PL/M	FORTRAN	FORTRAN BASIC COBOL
Residenter Assembler	✓	✓	✓
Monitor	✓	✓	✓

Bild 15.26
Mikroprozessor-Softwarebetrachtungen

KAPITEL 16

ZUKÜNFTIGE TRENDS

In diesem Kapitel werden die zu erwartenden technologischen Mikroprozessor-Entwicklungen im speziellen und die anstehenden technologischen Entwicklungen im allgemeinen untersucht und betrachtet; ferner werden die Trends auf dem Mikroprozessor-Markt diskutiert. Auch auf neue, mögliche System-Konfigurationen wird eingegangen. Schließlich wird dargelegt, wie sich der Mikroprozessor-Markt in den kommenden Jahren vermutlich entwickeln wird.

Zukünftige Trends

Dieses Kapitel verfolgt das Ziel, den Leser auf die zukünftigen Mikroprozessor-Entwicklungen vorzubereiten. Hierzu werden die Fortschritte der Mikroprozessor-Technologie beleuchtet, die Trends auf dem Mikroprozessor-Markt diskutiert und neue, mögliche System-Konfigurationen beschrieben. Darüber hinaus wird aufgezeigt, wie das Mikroprozessor-Marktangebot in den nächsten Jahren voraussichtlich aussehen wird.

Der bisherige Teil des Buches hatte die Aufgabe, dem Leser ein abgerundetes Wissen über die bestehende Mikroprozessor-Technologie zu vermitteln. Mit diesem letzten Kapitel wird versucht, ihn auf die zukünftigen Mikroprozessor-Entwicklungen vorzubereiten, indem auf die Trends der Mikroprozessor-Technologie eingegangen wird.

Diskussionsthemen
(Bild 16.1)

In diesem Kapitel werden die Trends auf dem Mikroprozessor-Markt untersucht (Bild 16.1). Danach werden die in den kommenden Jahren zu erwartenden technologischen Trends erörtert. Durch die mehr und mehr systembezogene Bauelemente-Entwicklung ergeben sich neue, mögliche Systemkonfigurationen, die ebenso beschrieben werden wie einige der gegenwärtigen Systeme, die auch in der Zukunft Bestand haben. Zum Schluß wird analysiert, wie – ausgehend von heute – in den nächsten 2 bis 5 Jahren das Mikroprozessor-Marktangebot aussehen könnte.

Stückzahl/Preis-Abhängigkeit von Halbleiterelementen
(Bild 16.2)

Dieser Blick in die Zukunft beginnt mit einer Untersuchung der Markt-Tendenzen, wobei am Anfang eine Analyse der klassischen Halbleiterbauelemente-Preiskurve (Preislernkurve) entsprechend Bild 16.2 steht. Während der vergangenen Jahre sind die Halbleiterprodukte ohne Ausnahme dem Verlauf dieser Kurve gefolgt. Sie bringt ganz einfach zum Ausdruck, daß – je mehr Bauelemente produziert werden – der Preis in einer definierten, bestimmbaren Weise abfällt. Wenn nun auch die Mikroprozessor-Preise entsprechend dieser Kurve weiter sinken werden, können 1-Chip-Mikrocomputer-Bausteine im Bereich von 10 DM und ganze Baustein-Sätze für leistungsfähigere Mikrocomputersysteme für unter 100 DM erwartet werden. Die Folge davon wird sein, daß der Mikroprozessor explosionsartig in alle Bereiche der Elektronik eindringen wird.

Aber zusätzlich zu dieser von der Lernkurve abhängigen Preisreduzierung gibt es noch weitere Gründe für diese eindrucksvolle Marktvorhersage.

- Trends des Mikroprozessor-Marktes
- Technologische Trends
- Systemkonzepte
- Voraussichtliches Mikroprozessor-Marktangebot

Bild 16.1
Diskussionsthemen

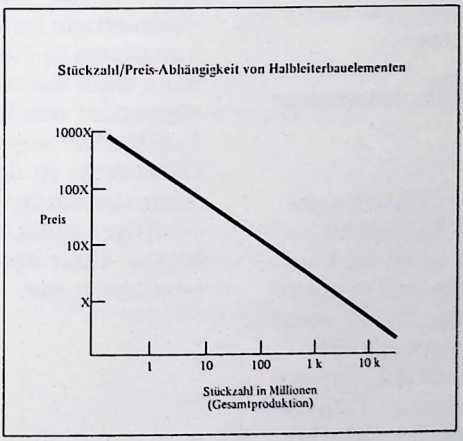


Bild 16.2
Stückzahl/Preis-Abhängigkeit von Halbleiterbauelementen

Mikroprozessor-Umsatz im Computermarkt
(Bild 16.3)

Einer 1974 veröffentlichten Übersicht [1] zufolge konnten Computer-Systeme von 1974 auf 1975 einen Umsatzzuwachs von nahezu $ 3 Mrd. erreichen. Man kann mit Sicherheit sagen, daß hier ein großer Markt besteht und daß beträchtliche Investitionssummen dafür ausgegeben werden, in diesen Markt sowohl von der Bauelemente- wie auch von der Systemseite her vorzudringen.
Außerdem lassen die bisher behandelten Geräte und Anlagen noch nicht den potentiellen Bedarf an Mikroprozessoren in einfachen Steuerungsanwendungen zum Einbau in andere Systeme erkennen. Allein dieser Marktbereich könnte der größte Abnehmer von Systemen auf Mikroprozessor-Basis sein.

Voraussichtliches Mikroprozessor-Wachstum
(Bild 16.4)

Alle Mikroprozessor-Markttendenzen deuten Wachstum und fortgesetzte, eifrige Entwicklungstätigkeit an. Für den Entwickler und Anwender bedeutet dies, daß sie auf Jahre hinaus weder weniger Produkt-Neuentwicklungen noch einen abnehmenden Mikroprozessor-Markt zu erwarten haben. Tatsächlich sagen mehrere Marktforscher für die nächsten fünf Jahre eine Verdoppelung des Marktes pro Jahr voraus (Bild 16.4). Unsere Daten sind nicht ganz so optimistisch. Allerdings sollten Sie auch nach unserer Meinung in den kommenden Jahren mit einem hektischen, schnell fortschreitenden Markt-Wachstum rechnen. Einige Mikroprozessor-Trends hängen sicherlich auch von den Bemühungen auf dem Gebiet der Bauelemente-Fertigungstechnologie ab.

Digitalsystem-Maßstab
(Bild 16.5)

Einer der wichtigsten Maßstäbe für den Entwickler von Digitalsystemen einerseits und den Anwender solcher Systeme andererseits ist durch einen Faktor gekennzeichnet, der den Fortschritt der Leistungsfähigkeit und der Kosten eines digitalen Systems in Beziehung setzt. Dieser Maßstabsfaktor, ausgedrückt in "Bits pro Sekunde pro DM oder Dollar", betrifft nicht nur das Gesamtsystem, sondern auch die Bauteile für dieses System. In unserem Fall sollen zwei Merkmale, die Leistung und die Kosten, als Maßstab dienen (Bild 16.5). Auf beide wird nachfolgend eingegangen, wobei mit der in "Bits pro Sekunde" gemessenen Leistung begonnen wird.

Will man die Trends der Bauelemente-Technologie bewerten, die zu immer komplexeren Bauelementen beiträgt, kann auch ein einfacherer Gütefaktor verwendet werden, der sich nur auf ein Logikgatter anwenden läßt. Ein solcher Gütefaktor ist das Geschwindigkeit x Verlustleistungs-Produkt, das bereits als wichtiger Indikator für die Leistungsfähigkeit einer bestimmten Technologie benützt wurde.

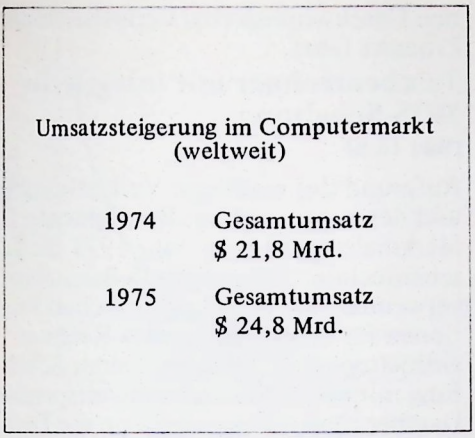


Bild 16.3
Umsatzsteigerung im Computermarkt

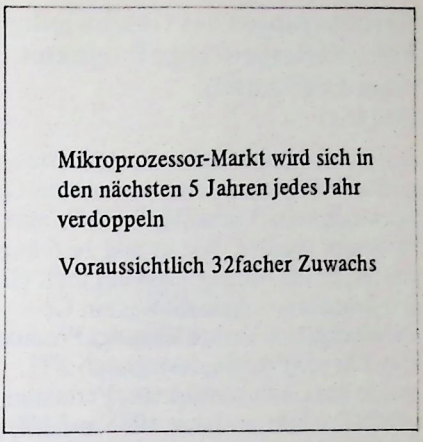


Bild 16.4
Voraussichtliches Mikroprozessor-Wachstum

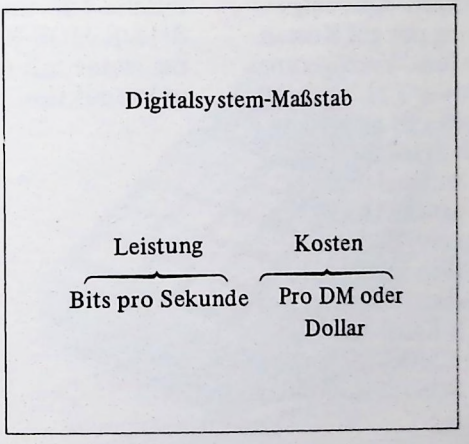


Bild 16.5
Digitalsystem-Maßstab

Verbesserungen des Geschwindigkeits/Verlustleistungs-Produktes eines Logikgatters
(Bild 16.6)

Aufgrund beträchtlicher Fortschritte innerhalb der letzten Jahre konnte das Geschwindigkeits/Verlustleistungs-Produkt verbessert werden. Wie in Bild 16.6 angedeutet, besaß Anfang 1963 ein DTL (Dioden-Transistor-Logik-)Gatter ein Geschwindigkeits/Verlustleistungs-Produkt von 200 pJ (Piko-Joules). Durch TTL wurde das Geschwindigkeits/Verlustleistungs-Produkt im Jahre 1965 auf 100 pJ herabgesetzt; eine weitere Reduzierung auf 60 pJ erfuhr es in der Zeit zwischen 1967 und 1970 durch ECL und Schottky-TTL. Damit etablierte sich ein Trend und als einziger bipolarer Technologie gelang es Low-Power TTL vor 1972, diesen Wert zu verbessern. Allerdings war diese Verbesserung nur auf Kosten der Geschwindigkeit bzw. Verzögerungszeit möglich. Low-Power TTL war mit einer Verzögerungszeit von $t_p = 30$ ns dreimal langsamer als Standard-TTL, aber es verbrauchte zehnmal weniger Leistung. Seit 1970 hat die MOS-Technik das Geschwindigkeits/Verlustleistungs-Produkt erheblich herabgesetzt. Der wichtigste Fortschritt kam erst mit P-Kanal-MOS, dann N-Kanal-MOS; anschließend folgte die CMOS-Technik und zum Schluß die SOS (Silicon-on-sapphire-)Technik.

Leistungsfähigkeit verschiedener Technologien
(Bild 16.7)

Um die Situation der Technologie besser zu überblicken, sind in Bild 16.7 die verschiedenen Technologien miteinander verglichen. Auch wenn die MOS-Technik – gekennzeichnet durch t_p – eine geringe Geschwindigkeit aufweist, so ist doch die Verlustleistung P_D sehr niedrig, was schließlich zu einem kleinen Geschwindigkeits/Verlustleistungs-Produkt führt.

Taschenrechner mit integrierter MOS-Schaltung
(Bild 16.8)

Aufgrund der niedrigen Verlustleistung und der angemessenen Bauelemente-Merkmale wurden im Jahr 1971 für Taschenrechner P-Kanal-MOS-Bausteine verwendet, die alle elektronischen Funktionen für einen Vierspezies-Rechner enthielten (Bild 16.8) und einer Schaltung mit 6000 Transistoren entsprachen. Darüber hinaus kennzeichnen die Fertigung von 1-K-MOS-Schreib-/Lesespeichern (RAM's) und 4-K-MOS-Festwertspeichern (ROM's) sehr deutlich den Trend nach höherer Funktions-Packungsdichte. Dieser Trend setzt sich weiter fort; bereits heute befinden sich dynamische 4-K- und 16-K-MOS-RAM's und 8/16-K-MOS-ROM's und sogar Speicherbausteine mit einer Kapazität von 32 K in Produktion.

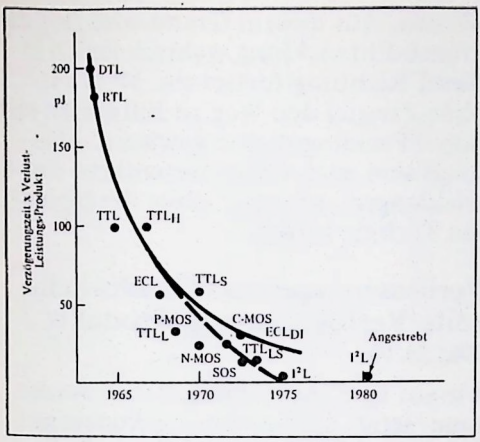


Bild 16.6
Verbesserungen des Geschwindigkeits/Verlustleistungs-Produktes eines Logikgatters

Jahr	Logikart	t_p (ns)	P_D (mW)	Geschwind./Verlust-Produkt (pJ)
	TTL			
1965	TTL	10	10	100
1967	TTL_H	5	20	100
1968	TTL_L	30	1	30
1970	TTL_S	3	20	60
1972	TTL_{LS}	10	2	20
1975	TTL_{LS}	5	2	10
	ECL			
1967	ECL_{10K}	2	30	60
1974	ECL_{DI}	0,7	43	30
	MOS			
1970	PMOS	200	0,1	20
1973	NMOS	100	0,1	10
1973	CMOS	*30	*1,0	*30
1974	SOS	*15	*0,5	*7,5
	*1 MHz			
	I^2L			
1975	$I^2L(5)$	35	0,085	3,0
1976	$I^2L(6)$	20	0,05	1,0
1980	$I^2L(0)$	10	0,10	1,0

Bild 16.7
Leistungsfähigkeit verschiedener Technologien

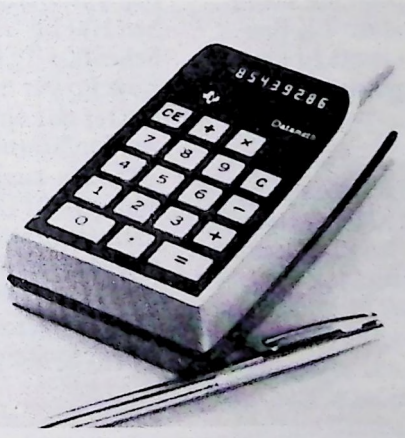


Bild 16.8
Taschenrechner mit integrierter MOS-Schaltung

Leistungsfähigkeit verschiedener Technologien
(Bild 16.9)

Wie aus Bild 16.9 zu entnehmen ist, verfügen bipolare Elemente über die höchste Geschwindigkeit bzw. kürzeste Verzögerungszeit. Für eine Verzögerungszeit t_p unterhalb 30 ns ist die Technologie ausschließlich bipolarer Natur; sie beginnt mit Low-Power TTL bei 30 ns und reicht herab bis auf 3 ns für Schottky TTL und 2 ns für ECL. Der Preis, der dafür bezahlt werden muß, ist die höhere Verlustleistung. ECL verbraucht 300 mal mehr Leistung als P- und N-Kanal-MOS. Zusammenfassend läßt sich feststellen, daß die Chip-Packungsdichte bei bipolaren Elementen etwa ein Viertel von der beträgt, die mit der MOS-Integration erreichbar ist. Das heißt, wo MOS mit einem 4-K-Speicher aufwarten kann, schafft man es bipolar nur bis zu einem 1-K-Speicher. Und ein 1-K-Entwurf in Schottky TTL entspricht hinsichtlich seiner Packungsdichte einem 256-Bit-Entwurf in ECL.

Man beachte, daß CMOS und SOS bei 1 MHz über ausgezeichnete Geschwindigkeits/Verlustleistungs-Produkte verfügen, die im bipolaren Gebiet liegen. Allerdings hat, wie später noch erläutert wird, CMOS nur ein Drittel der Packungsdichte von P- oder N-Kanal-MOS. Demgegenüber kann SOS 2- bis 3 mal so komplex sein wie P- oder N-Kanal-MOS. Gleichzeitig kann es 2 mal so schnell sein wie Standard -CMOS.

Die Verlustleistung von CMOS und SOS ändert sich mit der Frequenz. Deshalb und aufgrund der geringeren Packungsdichte liegen die CMOS-Anwendungen dort, wo geringere Geschwindigkeit und niedrige Verlustleistung wünschenswert sind. Der Integrationsgrad befindet sich auf mittleren Niveau. SOS kann, trotz $P_D = C \cdot U^2 \cdot f$, bei guten Geschwindigkeits-Eigenschaften für Entwicklungen mit hoher Packungsdichte verwendet werden. Aus diesem Grund wird sich die Prozeß-Entwicklung wahrscheinlich in dieser Richtung fortsetzen. MOS hat ohne Zweifel den Weg zu Entwürfen mit hoher Packungsdichte gewiesen. Allerdings sind auch einige wesentliche Entwicklungen auf dem Gebiet der bipolaren Technik erfolgt.

Verbesserungen des Geschwindigkeits/Verlustleistungs-Produkts
(Bild 16.10)

Wie aus Bild 16.10 entnommen werden kann, setzt die neu hinzugekommene dielektrisch isolierte ECL-Technik die Trendlinie fort. Die erste wirkliche Abweichung von dieser Kurve hat im Jahr 1972 mit der Entwicklung von Low Power Schottky TTL ein, dessen Geschwindigkeits/Verlustleistungs-Produkt 20 pJ beträgt. Die Packungsdichten haben sich derart erhöht, daß dort, wo bisher 100 bis 200 Gatter mit Standard-Schottky TTL möglich waren, nun mit Low Power Schottky TTL 500 bis 1000 Gatter auf einem Chip untergebracht werden können. Außerdem hat LPS-TTL den zusätzlichen Vorteil, mit den anderen TTL-Schaltungen direkt kompatibel zu sein, wodurch die Anpassung zwischen den Bausteinen relativ einfach ist. Eine solche Integration reduziert die Zahl der erforderlichen Bausteine in einem System um den Faktor 4.

Jahr	Logik-art	t_p (ns)	P_D (mW)	Geschwind./ Verlustl.-Produkt (pJ)
	TTL			
1965	TTL	10	10	100
1967	TTL_H	5	20	100
1968	TTL_L	30	1	30
1970	TTL_S	3	20	60
1972	TTL_{LS}	10	2	20
	ECL			
1967	ECL_{10K}	2	30	60
1974	ECL_{DI}	0,7	43	30
	MOS			
1970	PMOS	200	0,1	20
1973	NMOS	100	0,1	10
1973	CMOS	*30	*1,0	*30
1974	SOS	*15	*0,5	*7,5
	*1 MHz			
	I^2L			
1975	$I^2L(5)$	35	0,085	3,0
1976	$I^2L(6)$	20	0,05	1,0
1980	$I^2L(0)$	10	0,10	1,0

Bild 16.9
Leistungsfähigkeit verschiedener Technologien

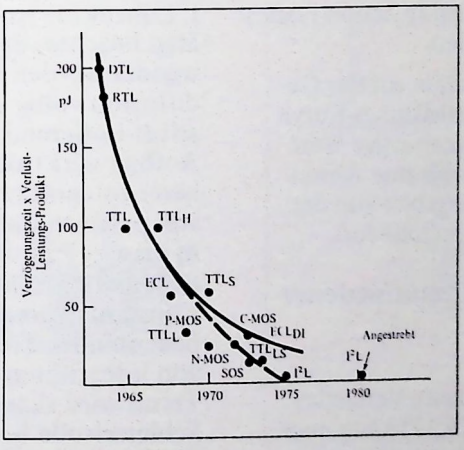


Bild 16.10
Verbesserungen des Geschwindigkeits/Verlustleistungs-Produkts

Minicomputer 960A von Texas Instruments
(Bild 16.11)

Durch die Fertigung von Einchip-Schottky-Akkumulatoren und Mikroprozessorelementen lassen sich heute Minicomputer der Klasse des TI 960-980 (Bild 16.11) mit weniger als 30 Bausteinen aufbauen. Unter Benutzung dieser vielseitigen Technologie wird sich die Integration signifikant fortsetzen, um mit Bauelementen extrem hoher Packungsdichte, den sog. VLSI (Very Large-Scale Integration)- und GSI (Grand-Scale Integration)-Schaltungen, die Familie an Hilfs- und Peripherie-Bausteinen auszufüllen.

Bis vor zwei Jahren hatte die Bipolar- gegenüber der MOS-Technologie hinsichtlich der Packungsdichte keine Chance. Letztere wird aber jetzt, wie man aus früheren Diskussionen weiß, durch die bipolare I^2L (Integrated Injection Logic)-Technik herausgefordert.

Man beachte ihre Position auf der Geschwindigkeits/Verlustleistungs-Kurve in Bild 16.10. Der gegenwärtige Wert von 3 pJ setzt die signifikante Abweichung auf dem Bipolargebiet von der normalen $t_p \times P_D$-Trendlinie fort.

Leistungsfähigkeit verschiedener Technologien
(Bild 16.12)

Aus der Geschwindigkeits/Verlustleistungs-Tabelle (Bild 16.12) kann man ersehen, daß I^2L gegenwärtig eine t_p von 35 ns und eine P_D von 85 μW pro Gatter aufweist. Das sind nicht unbedingt eindrucksvolle Geschwindigkeits-Eigenschaften, aber das daraus resultierende $t_p \times P_D$-Produkt ist 6 mal besser als von den anderen bipolaren Technologien und mehr als 2 mal besser als von der besten MOS-Technologie. **Und das ist erst der Anfang.**

Vorhersagen deuten darauf hin, daß I^2L bis 1980 sein Geschwindigkeits/Verlustleistungs-Produkt um den Faktor 3 verringern wird, wobei die wichtigste Verbesserung kleinere Verzögerungszeiten sein werden.

Vergleich von NAND/NOR-Gattern mit vier Eingängen
(Bild 16.13)

Die bipolare Technologie besitzt die gewünschte Geschwindigkeit und die niedrige Verlustleistung. Wie steht es aber mit der Packungsdichte? Kommt sie an die von MOS heran? Hierzu gibt Bild 16.13 einen Vergleich der erforderlichen Siliziumfläche, die mit den verschiedenen Technologien für den Aufbau eines Gatters mit vier Eingängen benötigt wird.

I^2L-Struktur
(Bild 16.14)

Die in diesem Beispiel benützte einfache I^2L-Struktur ist in Bild 16.14 dargestellt. Man beachte, daß nur Transistoren verwendet werden und daß eine Isolationsdiffusion völlig entfällt. Die Struktur ist selbst-isolierend. Dieser relativ einfache Aufbau wirkt sich direkt zugunsten höherer Integrationsdichte aus. Der Ersatz von passiven durch aktive Bauelemente, in diesem Fall werden hochohmige Diffusionswiderstände durch Transistor-Konstantstromquellen ersetzt, ist ein bedeutender Trend bei der Entwicklung von integrierten Schaltungen. Dieser Trend wird sich fortsetzen und eine Schlüsselrolle bei höchstintegrierten VLSI- oder GSI-Schaltungen spielen.

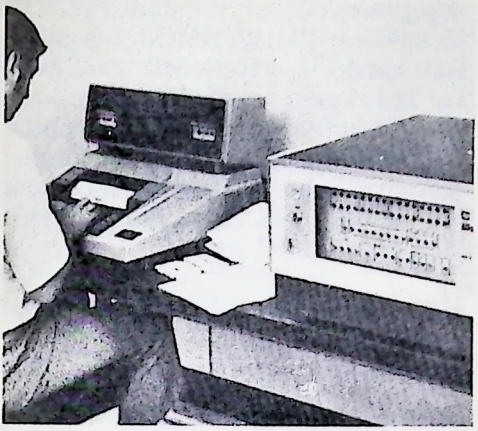


Bild 16.11
Minicomputer 960A von Texas Instruments

Jahr	Logik-art	t_p (ns)	P_D (mW)	Geschwind./ Verlustl.-Produkt (pJ)
	TTL			
1965	TTL	10	10	100
1967	TTL_H	5	20	100
1968	TTL_L	30	1	30
1970	TTL_S	3	20	60
1972	TTL_{LS}	10	2	20
1975	TTL_{LS}	5	2	10
	ECL			
1967	ECL_{10K}	2	30	60
1974	ECL_{DI}	0,7	43	30
	MOS			
1970	PMOS	200	0,1	20
1973	NMOS	100	0,1	10
1973	CMOS	*30	*1,0	*30
1974	SOS	*1,5	*0,5	*1,5
	*1 MHz			
	I^2L			
1975	$I^2L(5)$	35	0,035	3,0
1976	$I^2L(6)$	20	0,05	1,0
1980	$I^2L(0)$	15	0,10	1,0

Bild 16.12
Leistungsfähigkeit verschiedener Technologien

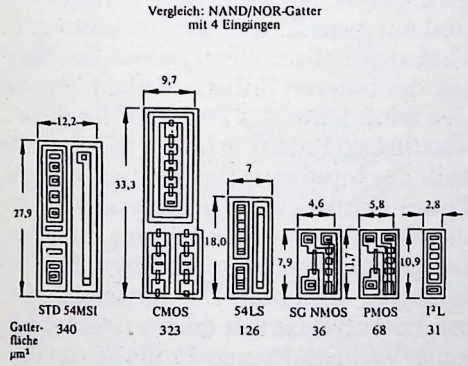


Bild 16.13
Vergleich von NAND/NOR-Gattern mit vier Eingängen

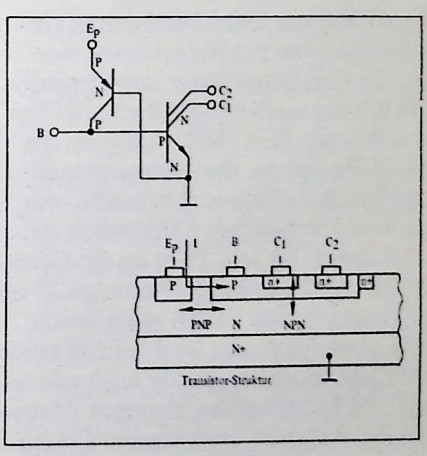


Bild 16.14
I^2L-Struktur

Die Ausbeute bestimmende Faktoren
(Bild 16.15)

Wenn man sich an unseren Maßstab zur Beurteilung von Digitalsystemen und Bausteinen erinnert, der in Bits pro Sekunde pro DM oder Dollar angegeben wird, müssen wir außer dem Faktor der Leistung noch einen zweiten Faktor in Betracht ziehen, nämlich den der Kosten. Ein Hersteller von integrierten Halbleiterbausteinen muß eine gute Ausbeute erzielen, um die "Bits pro Sekunde" zu einem vernünftigen Preis anbieten zu können. Eine gute Ausbeute hängt von den in Bild 16.15 angegebenen Punkten ab.

Gatter-Kosten gegenüber relativer Chipfläche
(Bild 16.16)

Ausgehend von festen bestimmten Herstellungskosten pro Waferscheibe werden die Kosten pro Gatter um so niedriger, je mehr Gatter man auf einem Chip unterbringen kann (Bild 16.16). In der Weise wie sich der Herstellungsprozeß und die Technologie stabilisierten, war ein Trend zu größeren Chipflächen zu verzeichnen. Im Jahr 1974 lag die Grenze der Chipfläche bei 6,25 mm x 6,25 mm; sie verschiebt sich jedoch stetig weiter nach oben. Trotzdem wird bei den größeren Chipflächen die gleiche Ausbeute erzielt wie früher bei den kleineren Abmessungen. Aufgrund der kleineren Fläche pro Gatter ergeben sich jetzt mehr Gatter pro Chip.

Erforderliche Masken und Diffusionsschritte für unterschiedliche Technologien
(Bild 16.17)

Die Einfachheit eines Prozesses läßt sich an der Zahl der Masken und Diffusionsschritte des jeweiligen Prozesses erkennen. In Bild 16.17 sind die Schritte angegeben, die für Standard TTL, CMOS, Metall-Gate-PMOS, NMOS (Silizium-Gate) und I^2L erforderlich sind. Aus diesen Daten geht zweifellos hervor, warum Metall-Gate-P-Kanal-MOS immer noch – bezogen auf die Leistung – ein sehr wirtschaftlicher Prozeß ist und warum I^2L ein solch gewaltiges Potential für die Zukunft darstellt.

Maschinen-Technologie gegenüber Applikation
(Bild 16.18)

Zusammenfassend kann schließlich für zukünftige Hochleistungs-Digitalsysteme, die mit VLSI- und GSI-Schaltungen arbeiten, folgendes erwartet werden:
Im MOS-Bereich wird P- und NMOS bei Anwendungen mit niedrigeren Geschwindigkeiten eingesetzt. Bei höheren Geschwindigkeiten wird SOS benützt, und zwar wegen der kurzen bipolaren Verzögerungszeit, der niedrigen Verlustleistung und der gegenüber PMOS zweifachen Packungsdichte. Wenn jedoch der Nachteil des teueren Substrats nicht überwunden wird, kann das Potential für diese Technologie nicht erfaßt werden. Innerhalb des bipolaren Bereichs bietet Low Power Schottky die höchste Geschwindigkeit, allerdings nur für den mittleren Integrationsgrad. Bei höchstem Integrationsgrad wird man I^2L verwenden. Dessen Potential ist mit einem Geschwindigkeits/Verlustleistungs-Produkt von 0,1 bis 1,0 pJ sicherlich sehr bedeutend, und I^2L kann letztlich MOS übertreffen. Und zwar vor allem deshalb, weil I^2L nicht nur über bipolare Geschwindigkeit und Treiber-Fähigkeit verfügt, sondern sich auch durch hohe Packungsdichte auszeichnet. Darüber hinaus kann I^2L auch bei sehr niedrigen Geschwindigkeiten mit proportional herabgesetztem Strom- und Leistungsbedarf betrieben werden, wobei das sehr niedrige Geschwindigkeits/Verlustleistungs-Produkt über einen weiten Temperaturbereich

- Kleine Fläche
- Einfache Herstellung
- Wenig Defektstellen
- Realistische Leistungs-Spezifikationen

Bild 16.15
Die Ausbeute bestimmende Faktoren

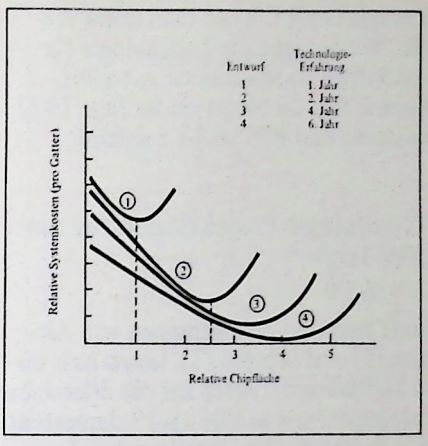


Bild 16.16
Relative Systemkosten pro Gatter in Abhängigkeit von der relativen Chipfläche

	STD-TTL	CMOS	P-MOS	SGN-MOS	I^2L
Masken	7	6	4	7	4
Diffusionsschritte	4	3	1	3	2

Bild 16.17
Erforderliche Maskenzahl und Diffusionsschritte für unterschiedliche Technologien

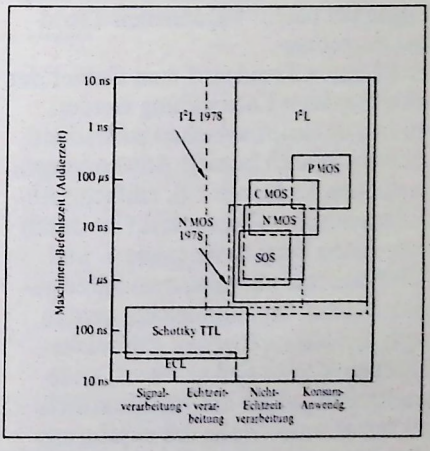


Bild 16.18
Maschinen-Technologie gegenüber Applikation

konstant bleibt. Wenn man unter diesen Bedingungen die Technologie für zukünftige Applikationen auswählt, könnte sich die Situation im Jahr 1977 entsprechend Bild 16.18 darstellen.

Technologie-Projektionen für die 80er-Jahre
(Bild 16.19)

Folgt man den Ausführungen von Altman [2] und Schmid [3], lassen sich diese Technologie-Trends auf die 80er-Jahre übertragen, wie in Bild 16.19 dargestellt. Was bedeuten diese Bauelemente-Technologietrends und -Grenzen für den Systementwurf? Sie sagen primär aus, daß man auf keinen Fall eine Revolution von neuen Bauelementen erwarten sollte, die auf irgendwelche eigentümliche Anwendungsprobleme abzielen. Vielmehr ist mit einer kontinuierlichen Entwicklung entlang der bereits begonnenen Trendlinie zu rechnen.
Die stärksten Trends auf dem Gebiet der Mikroprozessor-Entwicklung werden sich in drei Hauptbereichen auswirken. Der erste Bereich betrifft Anwendungen am unteren Ende, wie z.B. einfache Ablaufsteuerungen. Dieser Markt ist durch einen festen Befehlssatz sowie 4- und 8-Bit-Bausteine oder Baustein-Sätze gekennzeichnet. Im allgemeinen werden durch den Mikroprozessor zahlreiche Anwendungen zu Fall gebracht, andererseits wird jedoch die Wirtschaftlichkeit seiner Anwendung das explosionsartige Wachstum dieser Art von Applikationen diktieren.
Ein zweiter Trend wird sich in der Form bemerkbar machen, daß komplexe Bausteinfamilien mit einem festen Befehlssatz sowie einem 4-, 8- oder 16-Bit-CPU-Chip als Hauptmerkmal entwickelt werden. Diese Bausteinfamilien werden aufgrund eines gemeinsamen Bedarfs auf dem Markt mit einer gewissen Standardisierung der Funktionen angeboten.

Gegenwärtige Interface-Schaltungen
(Bild 16.20)

Als Beispiel zeigt Bild 16.20 die heute erhältlichen typischen Interface-Chips, die von einigen der notwendigen Detailentwicklungen entbinden. Diese Standard-Interfacebauelemente erfüllen als sog. Hilfs- oder Peripheriebausteine viele der Aufgaben, die einen ziemlich hohen Entwicklungsaufwand erfordern, wenn sie vom Hersteller nicht geliefert werden. Solche Dinge wie Fehlerprüfung, Bit-Formatierung usw. sind in diesen Bausteinen bereits enthalten.

Nächste Interface-Generation
(Bild 16.21)

In der nahen Zukunft werden diese Standard-Interfacebausteine derart erweitert, daß sie dann z.B. Bildschirm- und Floppy Disc-Funktionen enthalten. Folglich wird unser Konzept eines zweiten, weiterführenden Haupt-Vorstoßes darin bestehen, den Mikroprozessor-Markt kräftig mit Ein-/Ausgabebausteinen zu unterstützen, kombiniert mit einer Vielzweck-CPU festen Befehlssatzes für eine vielseitige Bausteinfamilie. Demgemäß wird aufgrund des Marktbedarfs erwartet, daß Hersteller Standard-Funktionen anbieten müssen, wie z.B. eine gemeinsame Interface-Einrichtung. Eine Standardisierung der Pin-Konfigurationen oder Befehlssätze durch die Hersteller wird dagegen nicht erwartet.
Ein dritter Haupt-Vorstoß der Mikroprozessor-Entwicklung wird sich auf das Gebiet der fortgeschrittenen Architektur und Betriebssysteme erstrecken, die es dem anspruchsvolleren Mikroprozessor erlauben, so effektiv wie die fortgeschritteneren Digitalsysteme von heute zu arbeiten. Diese Entwicklungen werden in drei Basisbereichen stattfinden.

Projektionen für die 80er-Jahre

LSI-Parameter	1973/1974 LSI	1980's (VLSI)
Leistung		
Max. Taktfrequenz (MHz)	30	100
Geschwindigkeits/Verlust-leistungs-Produkt (pJ)	3 bis 10	0,1 bis 1
Komplexität		
Max. Chipfläche (Kantenlänge in mil)	250	500
Transistorfläche (mil^2)	1 bis 2	0,1 bis 0,3
Gatterfläche (mil^2)	5 bis 10	1,0 bis 3,0
Bitfläche (mil^2)	2	0,5
Bauelementedichte (Transisturen/Chip)	5000	200.000
Funktionen		
Schreib-/Lesespeicher	4096 Bits	64.000 Bits
(Random) Logik	500 Gatter	10.000 Gatter
Minicomputer-CPU (bipolar)	10 Chips bei 5 MHz	1 Chip bei 50 MHz

Bild 16.19
Technologie-Projektionen für die 80er-Jahre

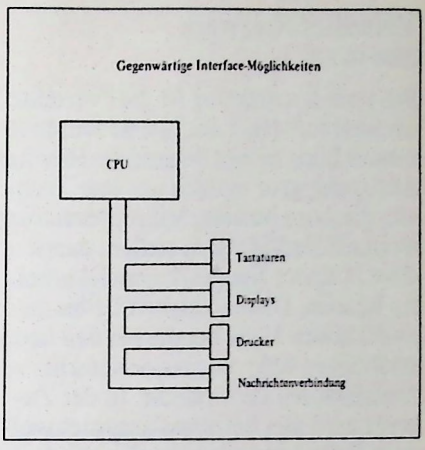


Bild 16.20
Gegenwärtige Interface-Möglichkeiten

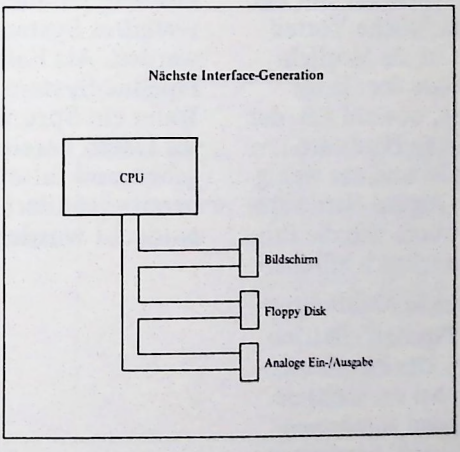


Bild 16.21
Nächste Interface-Generation

Verteiltes Netzwerk
(Bild 16.22)

Die erste Entwicklung ist die "verteilte Architektur" (Bild 16.22); sie wurde in diesem Buch bereits behandelt. Hier handelt es sich ganz einfach um eine Technik, die darin besteht, Mikroprozessoren Wege zur Verfügung zu stellen, damit diese in einem Netzwerk parallel arbeiten können. Diese Fähigkeit ist bis zu einem hohen Maße bereits bei den heute erhältlichen Mikroprozessorelement-Architekturen verwirklicht. In der Zukunft wird dies bei etwas anspruchsvolleren Betriebssystemen noch einfacher zu erreichen sein.
Eine zweite Architektur/Betriebssystem-Entwicklung wird wahrscheinlich als virtuelles System erhältlich sein. Bei einem virtuellen System weiß der Benutzer nichts von der Hardware, mit der er arbeitet. Der offensichtliche Vorteil eines solchen Systems ist die Möglichkeit entwickelte Software über lange Zeiträume zu benutzen, obwohl z.B. der Prozessor mit einer neuen Hardware-Generation aufgebaut ist und nur wenig Ähnlichkeit mit dem Original-Hardwaresystem hat. Dadurch würde sich die Produkt-Lebensdauer beträchtlich erhöhen.

Eine dritte, zu erwartende Architektur-Entwicklung ist der "Pipeline"-Betrieb von Mikroprozessoren. Der Pipeline-Betrieb ist eine Technik, bei der mehrere Prozessoren – in Gruppen zusammengefaßt – ein Problem seriell bearbeiten und durch überlappende Funktionen den Maschinendurchsatz erhöhen. Die typischen fünf Schritte, die eine digitale Maschine auszuführen hat, sind: Hole Befehl, decodiere Befehl, erzeuge Adresse, hole Daten und führe aus.

Pipeline-Architektur
(Bild 16.23)

Wie in Bild 16.23 dargestellt, kann ein bestimmter Prozessor die Aufgabe zugeteilt bekommen, Befehle zu holen und zu decodieren; ein anderer erzeugt die Adresse, während wiederum andere Prozessoren mit der Zahlenverarbeitung betraut sind.

Überlappender Betrieb
(Bild 16.24)

Mit einer Pipeline-Betriebsstruktur können Funktionen entsprechend Bild 16.24 überlappend ausgeführt werden. Man stelle sich vor, daß jede dieser Operationen an einer Stufe der Pipeline (Bild 16.23) eintritt, wobei der Datenfluß durch die Stufen erfolgt. Die unproduktive Zeit der Daten-Prozessoren würde dann ein Minimum betragen. Und der System-Durchsatz würde viele Male größer sein als mit einem einzelnen Prozessor. Natürlich bringt diese Strukturart serielle Probleme mit sich. Falls das Problem paralleler Natur ist, sollte ein *verteiltes* System in Betracht gezogen werden. Als Fallgrube stellt sich das Pipeline-System bei Sprüngen heraus. Wenn ein Sprung entdeckt wird, sind die Daten bereits an die Pipeline übergeben und tatsächlich können die Daten bereits geändert sein, bevor der Sprung entdeckt worden ist.

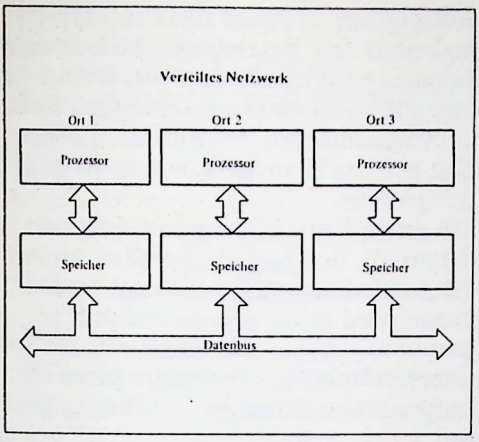


Bild 16.22
Verteiltes Netzwerk

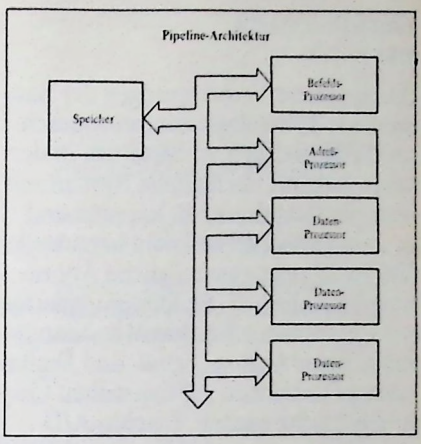


Bild 16.23
Pipeline-Architektur

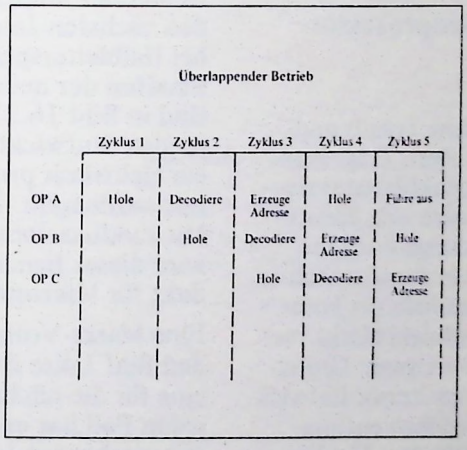


Bild 16.24
Überlappender Betrieb

Einchip-System
(Bild 16.25)

Durch weitere Verbesserungen der Bauelemente-Technologie erscheint durchaus die Möglichkeit zu bestehen, in der nahen Zukunft ein digitales System mit analogen Funktionen zu koppeln und als monolithischen Baustein herzustellen. Bild 16.25 zeigt eine mögliche Anordnung. In einfachen, für Steuerungsaufgaben vorgesehenen Einchip-Mikrocomputern können Analog-Digital- und Digital-Analog-Funktionen auf demselben Chip untergebracht werden. Einchip-A/D- und D/A-Umsetzer sind bereits erhältlich. Folglich wird die Entwicklung von Einchip-Systemen nicht mehr lange auf sich warten lassen und es ist vernünftig anzunehmen, daß solche Systeme in der nahen Zukunft verfügbar sein werden.

Bevorstehende Mikroprozessor-Entwicklungen
(Bild 16.27)

Bisher wurden zahlreiche Trends und Systemkonzepte diskutiert. Trägt man sie als bedeutende Produktdaten systematisch zusammen, lassen sich daraus verschiedene Entwicklungs-Vorstöße ableiten. So wird sich in der allernächsten Zukunft, also innerhalb der kommenden zwei Jahre, auf dem Markt voraussichtlich folgendes ereignen: Grundsätzlich werden Mikroprozessor-Entwicklungen in den drei Bereichen entsprechend Bild 16.27 stattfinden. Ein Entwicklungs-Vorstoß geht in Richtung in sich abgeschlossener, komplexer 8- oder 16-Bit-Einchipsysteme. Diese Bausteine zielen auf einfache Anwendungen hoher Stückzahl ab, die prinzipiell und in der Regel eindeutig definiert innerhalb des Steuerungs- und Nachrichtentechnik-Sektors zu finden sind.
Der zweite Entwicklungs-Vorstoß betrifft den Bereich der 8- und 16-Bit-Bausteinfamilien. Es wird sich zeigen, daß die Hersteller diese Familien als vollständige Systeme mit Prototyp-Fähigkeiten und Betriebssystem-Softwarepaketen entwickeln. Darüber hinaus wird hier aufgrund eines gemeinsamen Bedarfs an Anwendungen auf Funktionsebene eine gewisse Standardisierung zu beobachten sein.
Die dritte Entwicklungsrichtung zielt darauf ab, den anspruchsvollen Anwender zufriedenzustellen. Diesen Bedürfnissen wird in der nahen Zukunft am besten mit Anwender-definierbaren Softwaresystemen und kaskadierbaren Mikroprozessorelementen Rechnung getragen.

Nächste Generation an Speicherbausteinen
(Bild 16.27)

Zusätzlich zu den Fortschritten auf dem Gebiet der Mikroprozessoren wird es in den nächsten Jahren auch Fortschritte bei Halbleiterspeichern geben. Die Eigenschaften der neuen Speicherbausteine sind in Bild 16.27 angegeben. Die bedeutendste Entwicklung dürfte vermutlich ein elektrisch programmierbarer ROM mit wählbarem Einschreibmodul sein. Mit Zykluszeiten für ROM's von $< 1\,\mu s$ wird dieses Bauelement ein gängiges Produkt für Mikroprozessorsysteme werden.

Eine Markt-Vorhersage für die kommenden fünf Jahre ist ebenso schwierig wie eine für die nächsten zwei Jahre. Auf jeden Fall hat es den Anschein, daß man mit markanten Entwicklungstrends zu rechnen hat. Als erstes erscheint in diesem Zeitraum der 16-Bit-Computer auf einem Chip als mögliches Produkt. Kleinere Verbesserungen der NMOS- und I^2L-Technologie genügen bereits, um mit beiden Technologien einen solchen "Computer auf einem Chip" entwickeln zu können.

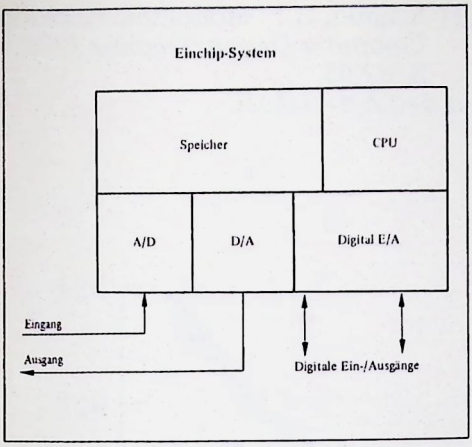


Bild 16.25
Einchip-System

Erster Entwicklungs-Vorstoß –
8- oder 16-Bit. In sich abgeschlossen,
komplexen Einchip-System

Zweiter Entwicklungs-Vorstoß –
8- und 16-Bit-Bausteinfamilien

Dritter Entwicklungs-Vorstoß –
Bedürfnisse des anspruchsvollen
Anwenders angepaßt

Bild 16.26
Bevorstehende Mikroprozessor-Entwicklungen

Nächste Generation an Speicherbausteinen

Typ	Kapazität	Organisation	Zykluszeit
Dyn. RAM	16 K	16384x1 od. 4096x4	<500 ns
Stat. RAM	4K	4096x1	<80 ns
EAROM	8/16 K	Block- oder Byte-Programmierung	<1 µs
EPROM	16 K	1024 x 8	<500 ns

Bild 16.27
Nächste Generation an Speicherbausteinen

Zusammenhang zwischen Größe und Kosten
(Bild 16.28)

Wie sieht es mit den Kosten in den kommenden Jahren aus? In Bild 16.28 ist der Verlauf der Kosten aufgetragen. Man kann daraus entnehmen, daß eine vollständige CPU auf einem Chip bald weniger als 25 DM kostet. Dies bedeutet für die Industrie, daß die Kosten für Computerleistung oder für die Hardware von anspruchsvollen Controllern vernachlässigbar werden. Zusätzlich zum Computer auf einem Chip erscheint in diesem Zeitraum die Verbindung von analogen und digitalen Technologien auf einem monolithischen Chip durchführbar. Man bedenke, was dies allein für den Peripherie-Markt bedeuten könnte.

Zusammenfassung

Aus diesem Kapitel sollten Sie hinreichende Kenntnisse über die Markttendenzen gewonnen haben, die bei Mikroprozessoren in Erscheinung treten. Ebenso sollten Sie in der Lage sein, die wichtigsten, demnächst zu erwartenden Vorstöße der Mikroprozessor-Hersteller zu verfolgen. Außerdem sollten Sie jetzt die 5-Jahres-Kostenprognosen und die technischen Grenzen von Mikroprozessor-Produkten realistisch einschätzen können.

Literatur

[1] Market Research Department of Computerworld Incorporated. *Computerworld Marketbrief (October 1974).*

[2] Altman, L.: "Logics' Leap Ahead Creates New Design Tools for Old and New Applications". *Electronics (February 21, 1974), S. 81-90.*

[3] Schmid, H.: "Monolithic Processors". *Computer Design (October 1974), S. 87-95.*

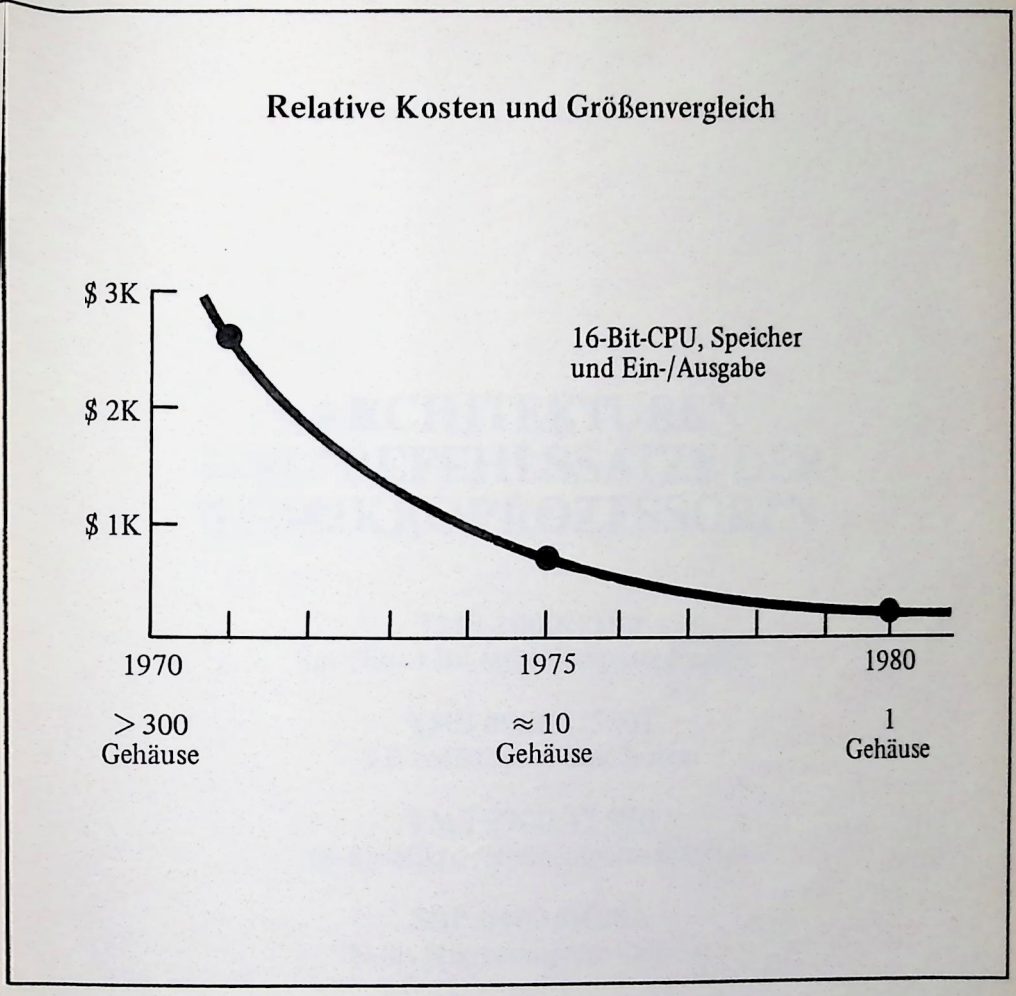


Bild 16.28
Relative Kosten und Größenvergleich

ARCHITEKTUREN UND BEFEHLSSÄTZE DER TI-MIKROPROZESSOREN

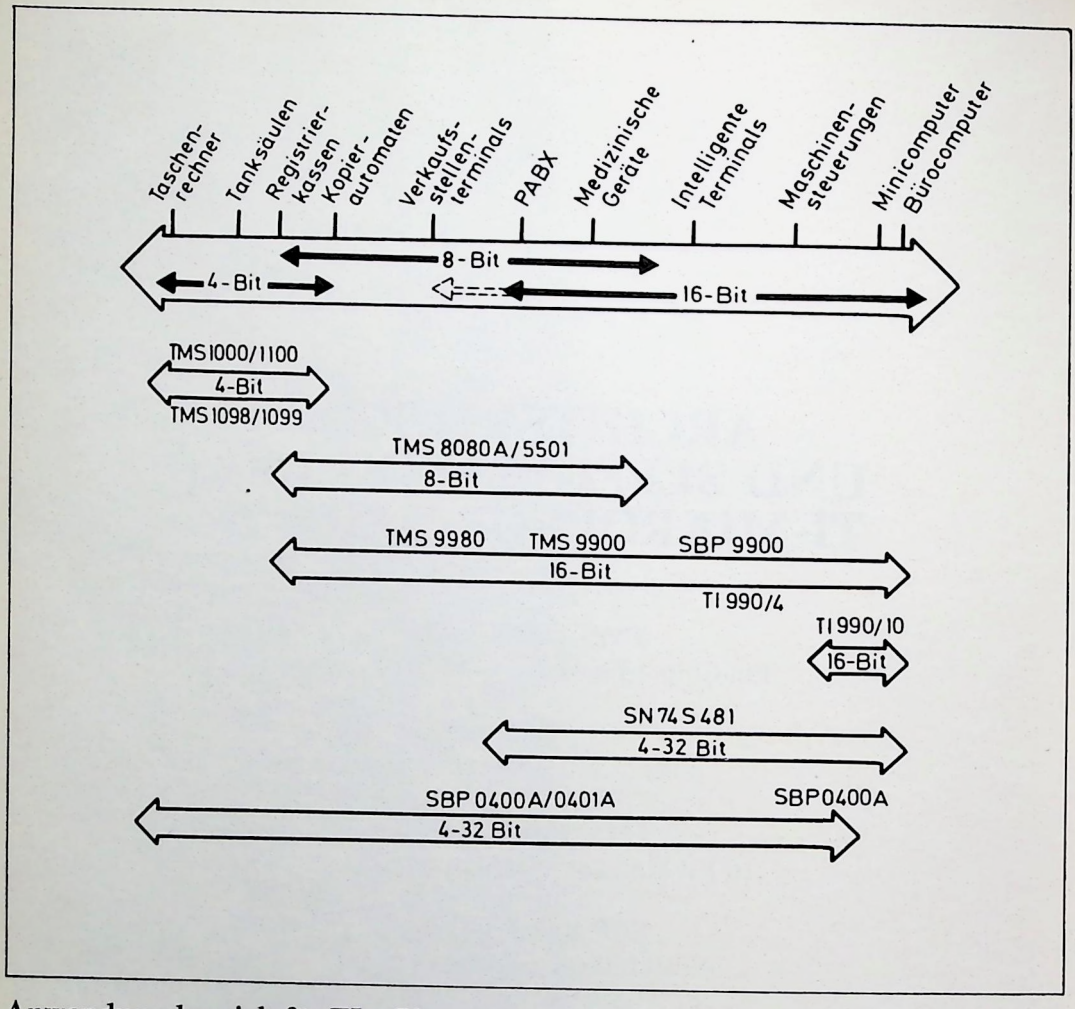
TMS 1000/1100
Ein-Chip 4-Bit-Mikrocomputer-Familie

TMS 8080A/5501
8-Bit-Mikroprozessor-System

TMS 9900/TI 990
16-Bit-Mikro-/Minicomputer-Konzept

SBP 0400/0400A
N-Bit Mikrocomputer Chip-Set

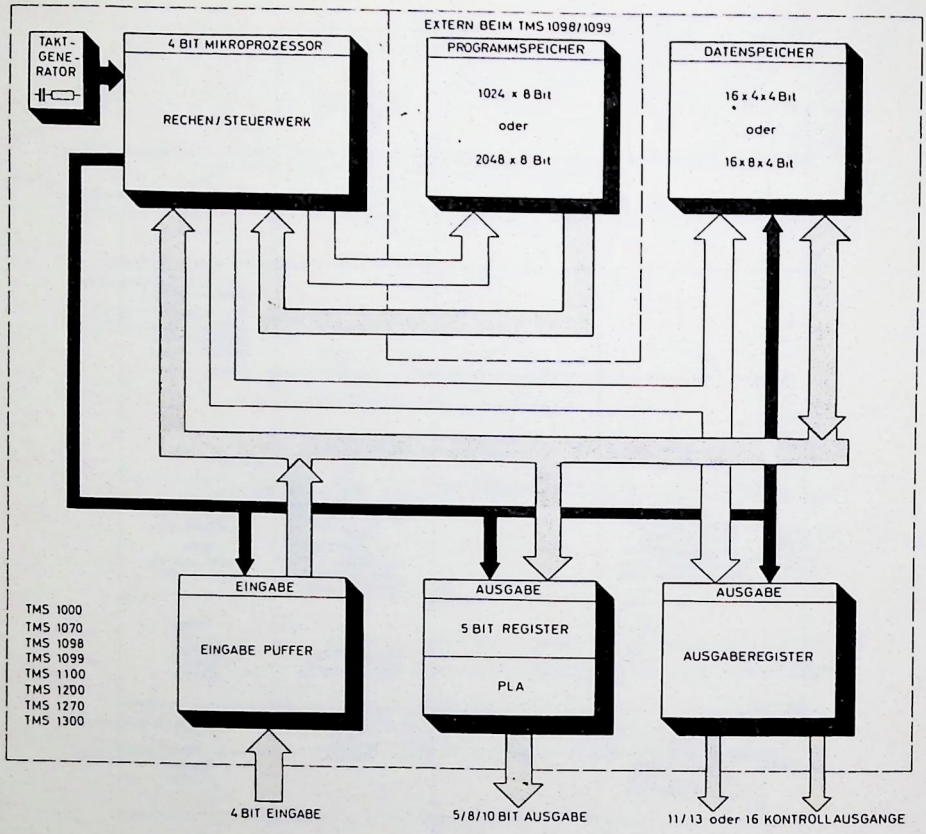
SN 74S481/S482
N-Bit Mikrocomputer Chip-Set



Anwendungsbereich für TI-μP's

Worte	Bit pro Wort			
	1	2	4	8
16	16		RAMs SN54S189/SN74S189 SN54S289/SN74S289 SN7489	
32	32	64		RAMs TMS4036 PROMs SN54185/SN74185
64	64	128		ROMs SN5488A/SN7488A PROMs SN54188A/SN74188A SN54S188/SN74S188 SN54S288/SN74S288
128	128	256		
256	RAMs SN54S201/SN74S201 SN54S301/SN74S301 SN74S200A/SN74S300A SN74LS200A/SN74LS300A SN74LS202A/SN74LS302A	512	RAMs TMS4039, TMS4042, TMS4043 SN74S207, SN74S208 ROMs SN54187/SN74187 PROMs SN54S287/SN74S287 SN54S387/SN74S387	ROMs SN54S271/SN74S271 SN54S371/SN74S371 PROMs SN54S470/SN74S470 SN54S471/SN74S471
512		1024	ROMs SN54S270/SN74S270 SN54S370/SN74S370	PROMs SN74S472 SN74S473 SN74S474 SN74S475
1024	RAMs TMS4033, TMS4034, TMS4035 TMS4062, TMS4063 SN74S214, SN74S314 SN74LS219, SN74LS314	2048	RAMs TMS 4045 TMS 4047	ROMs TMS4700 EROMs TMS2708 TMS 27L08
2048		4096		ROMs TMS4800 EROMs TMS 2716
4096	RAMs TMS4030, TMS4050 TMS4051, TMS4060 TMS 4044 TMS 4046	8192	ROMs TMS4830	ROM TMS4732
8192		16384		
16384	RAM TMS4070 TMS 4071	32768		

Speicherauswahlliste



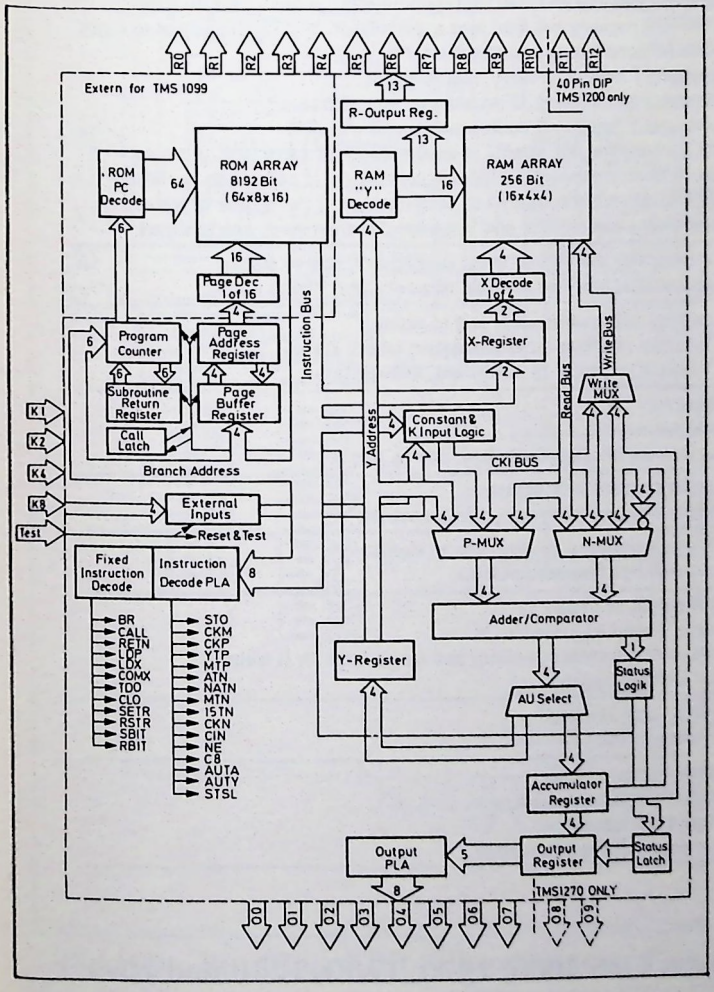
TMS 1000
TMS 1100 Ein-Chip 4-Bit-Mikrocomputer-Familie

Eigenschaften

- 8k-/16k-Bit-Programmspeicher
- 256-/512-Bit-Datenspeicher
- Interner Taktgenerator
- 4 parallele Eingänge
- 8 parallele Ausgänge mit PLA
- 11/13 oder 16 speichernde Kontrollausgänge
- 8-Bit-Befehlsformat
- 4-Bit-Datenformat
- 43/54 Standardbefehle
- Mikroprogrammierbar
- Befehlszykluszeit 15 μs min.
- Leistungsaufnahme 90/105 mW typ.

Typ	Programm-Speicher	Daten-Speicher	O-Ausgänge	R-Ausgänge	Gehäuse	Standard Befehlssatz
TMS1000	8192 Bit	256 Bit	8 (20 V)	11	28 Pin	TMS1000
TMS1070	8192 Bit	256 Bit	8 (35 V)	11	28 Pin	TMS1000
TMS1200	8192 Bit	256 Bit	8 (20 V)	13	40 Pin	TMS1000
TMS1270	8192 Bit	256 Bit	10 (35 V)	13	40 Pin	TMS1000
TMS1099	Extern	256 Bit	5 (20 V)	13	64 Pin	TMS1000 *
TMS1100	16384 Bit	512 Bit	8 (20 V)	11	28 Pin	TMS1100
TMS1300	16384 Bit	512 Bit	8 (20 V)	16	40 Pin	TMS1100
TMS1098	Extern	512 Bit	5 (20 V)	16	64 Pin	TMS1100 *

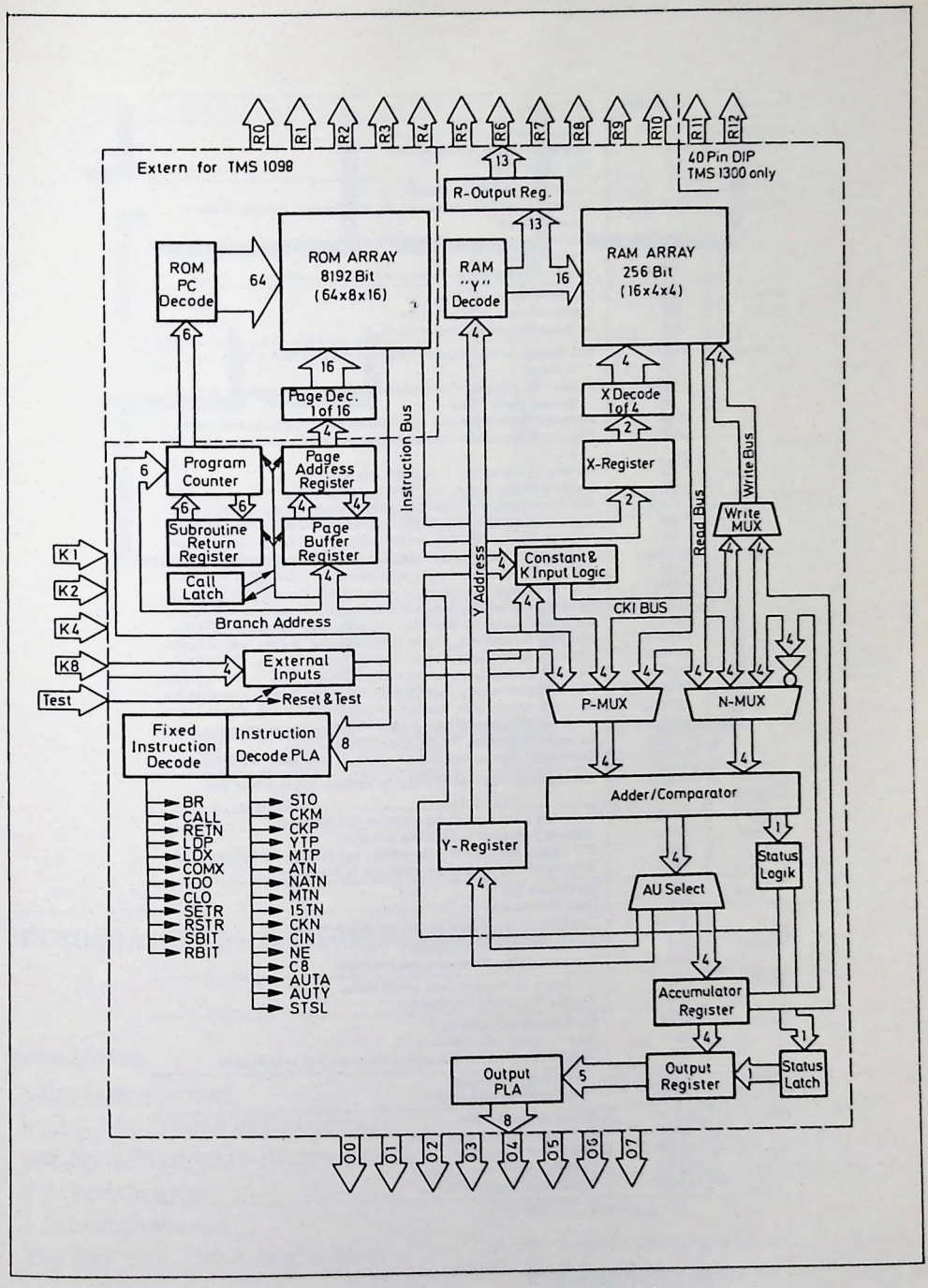
* Wird nur mit Standard-Befehlssatz geliefert



TMS 1000
Ein-Chip 4-Bit-Mikrocomputer-Familie

Function	MNEMONIC	Description
Register to Register	TAY	Transfer accumulator to Y register.
	TYA	Transfer Y register to accumulator.
	CLA	Clear accumulator.
Transfer Register to Memory	TAM	Transfer accumulator to memory.
	TAMIY	Transfer accumulator to memory and increment Y register.
	TAMZA	Transfer accumulator to memory and zero accumulator.
Memory to Register	TMY	Transfer memory to Y register.
	TMA	Transfer memory to accumulator.
	XMA	Exchange memory and accumulator.
Arithmetic	AMAAC	Add memory to accumulator, results to accumulator. If carry, one to status.
	SAMAN	Subtract accumulator from memory, results to accumulator. If no borrow, one to status.
	IMAC	Increment memory and load into accumulator. If carry, one to status.
	DMAN	Decrement memory and load into accumulator. If no borrow, one to status.
	IA	Increment accumulator, no status effect.
	IYC	Increment Y register. If carry, one to status.
	DAN	Decrement accumulator. If no borrow, one to status.
	DYN	Decrement Y register. If no borrow, one to status.
	A8AAC	Add 8 to accumulator, results to accumulator. If carry, one to status.
	A10AAC	Add 10 to accumulator, results to accumulator. If carry, one to status.
	A6AAC	Add 6 to accumulator, results to accumulator. If carry, one to status.
	CPAIZ	Complement accumulator and increment. If then zero, one to status.
Arithmetic Compare	ALEM	If accumulator less than or equal to memory, one to status.
	ALEC	If accumulator less than or equal to a constant, one to status.
Logical Compare	MNEZ	If memory not equal to zero, one to status.
	YNEA	If Y register not equal to accumulator, one to status.
	YNEC	If Y register not equal to a constant, one to status.
Bits in Memory	SBIT	Set memory bit.
	RBIT	Reset memory bit.
	TBIT1	Test memory bit. If equal to one, one to status.
Constants	TCY	Transfer constant to Y register.
	TCMIY	Transfer constant to memory and increment Y.
Input	KNEZ	If K inputs not equal to zero, one to status.
	TKA	Transfer K inputs to accumulator.
Output	SETR	Set R output addressed by Y.
	RSTR	Reset R output addressed by Y.
	TDO	Transfer data from accumulator and status latch to O outputs.
	CLO	Clear O-output register.
RAM 'X' Addressing	LDX	Load 'X' with a constant.
	COMX	Complement 'X'.
ROM Addressing	BR	Branch on status = one.
	CALL	Call subroutine on status = one.
	RETN	Return from subroutine.
	LDP	Load page buffer with constant.

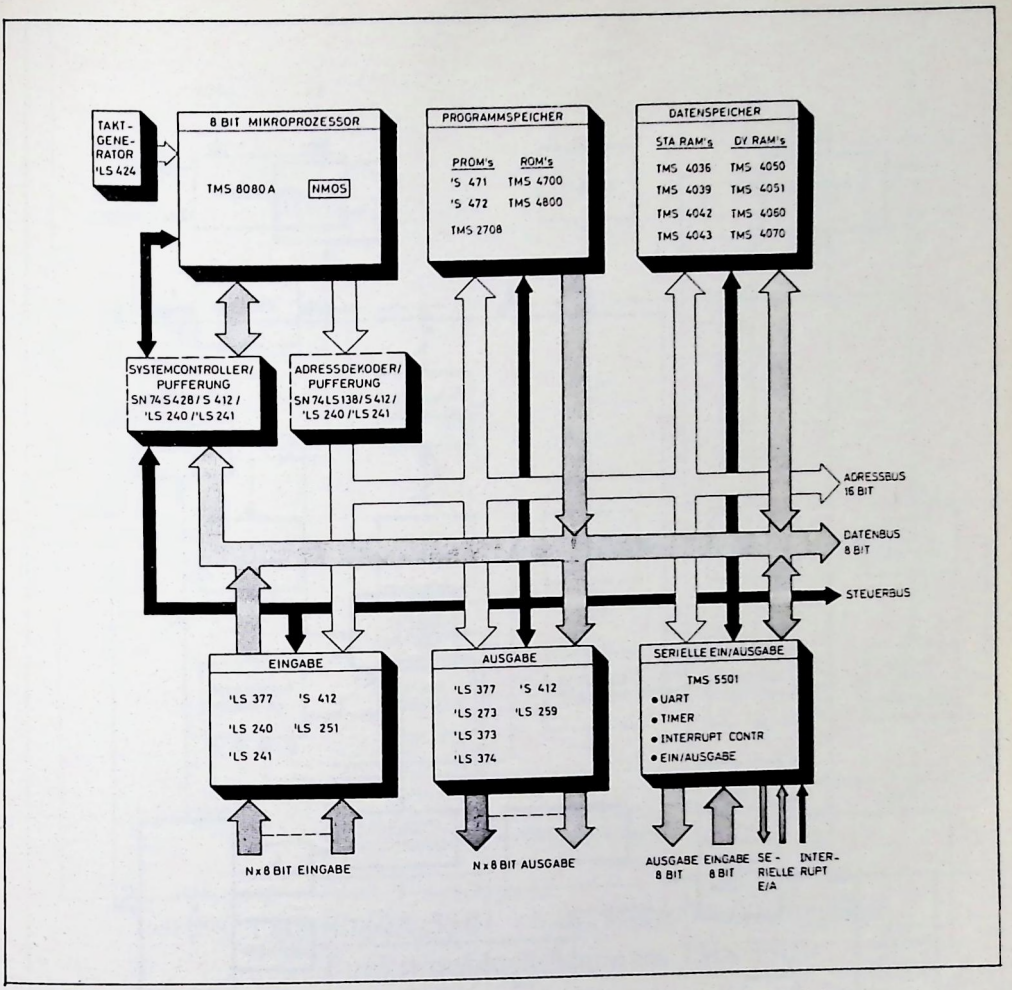
Standard Befehlsvorrat des TMS 1000, 1070, 1200, 1270 und 1099



TMS 1100 Ein-Chip 4 Bit Mikrocomputer-Familie
Funktionsblockdiagramm des TMS 1100, 1300 und 1098

Funktion	MNEMONIC	Beschreibung
Register-to-Register Transfer	TAY	Transfer Accumulator to Y-Register
	TYA	Transfer Y-Register to Accumulator
	CLA	Clear Accumulator
Register to Memory	TAM	Transfer Accumulator to Memory
	TAMIYC	Transfer Accumulator to Memory and increment Y-Register. If carry, one to Status
	TAMDYN	Transfer Accumulator to Memory and decrement Y-Register. If no borrow, one to Status
	TAMZA	Transfer Accumulator to Memory and zero Accumulator
Memory to Register	TMY	Transfer Memory to Y-Register
	TMA	Transfer Memory to Accumulator
	XMA	Exchange Memory and Accumulator
Arithmetic	AMAAC	Add Memory to Accumulator, Results to Accumulator. If carry, one to Status
	SAMAN	Subtract Accumulator from Memory, Results to Accumulator. If no borrow, one to Status
	IMAC	Increment Memory and load into Accumulator. If carry, one to Status
	DMAN	Decrement Memory and load into Accumulator. If no borrow, one to Status
	IAC	Increment Accumulator. If no carry, one to Status
	DAN	Decrement Accumulator. If no borrow, one to Status
	A2AAC	Add 2 to Accumulator, Results to Accumulator. If carry, one to Status
	A3AAC	Add 3 to Accumulator, Results to Accumulator. If carry, one to Status
	A4AAC	Add 4 to Accumulator, Results to Accumulator. If carry, one to Status
	A5AAC	Add 5 to Accumulator, Results to Accumulator. If carry, one to Status
	A6AAC	Add 6 to Accumulator, Results to Accumulator. If carry, one to Status
	A7AAC	Add 7 to Accumulator, Results to Accumulator. If carry, one to Status
	A8AAC	Add 8 to Accumulator, Results to Accumulator. If carry, one to Status
	A9AAC	Add 9 to Accumulator, Results to Accumulator. If carry, one to Status
	A10AAC	Add 10 to Accumulator, Results to Accumulator. If carry, one to Status
	A11AAC	Add 11 to Accumulator, Results to Accumulator. If carry, one to Status
	A12AAC	Add 12 to Accumulator, Results to Accumulator. If carry, one to Status
	A13AAC	Add 13 to Accumulator, Results to Accumulator. If carry, one to Status
	A14AAC	Add 14 to Accumulator, Results to Accumulator. If carry, one to Status
	IYC	Increment Y-Register. If carry, one to Status
	DYN	Decrement Y-Register. If carry, one to Status
	CPAIZ	Complement Accumulator and increment. If then Zero, one to Status
Arithmetic Compare	ALEM	If Accumulator less than - or equal to Memory, one to Status
Logical Compare	MNEA	If Memory is not equal to Accumulator, one to Status
	MNEZ	If Memory not equal to Zero, one to Status
	YNEA	If Y-Register not equal to Accumulator, one to Status and Status Latch
	YNEC	If Y-Register not equal to Constant, one to Status
Bits in Memory	SBIT	Set Memory Bit
	RBIT	Reset Memory Bit
	TBIT1	Test Memory Bit. If equal to one, one to Status
Constants	TCY	Transfer Constant to Y-Register
	TCMIY	Transfer Constant to Memory and increment
INPUT	KNEZ	If K-Inputs not equal to Zero, one to Status
	TKA	Transfer K-Inputs to Accumulator
Output	SETR	Set R-Output addressed by Y
	RSTR	Reset R-Output addressed by Y
	TDO	Transfer data from Accumulator and Status Latch to O-Outputs
RAM X Addressing	LDX	Load X with File Address
	COMX	Complement the MSB of X
ROM	BR	Branch on Status = One
	CALL	Call Subroutine on Status = One
	RETN	Return from Subroutine
	LDP	Load Page Buffer with Constant
	COMC	Complement Chapter Buffer

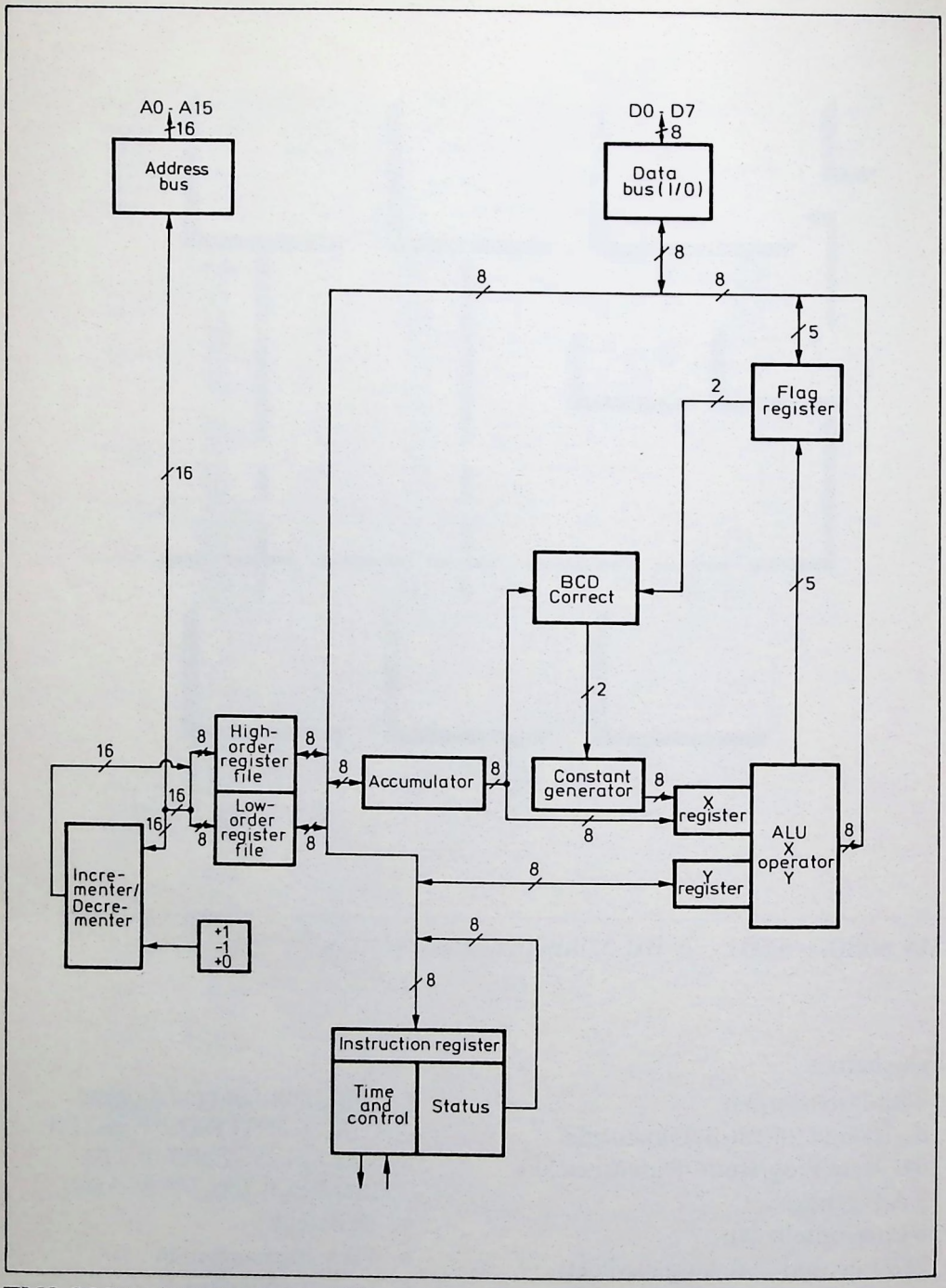
Standard-Befehlsvorrat des TMS 1100, 1300 und 1098



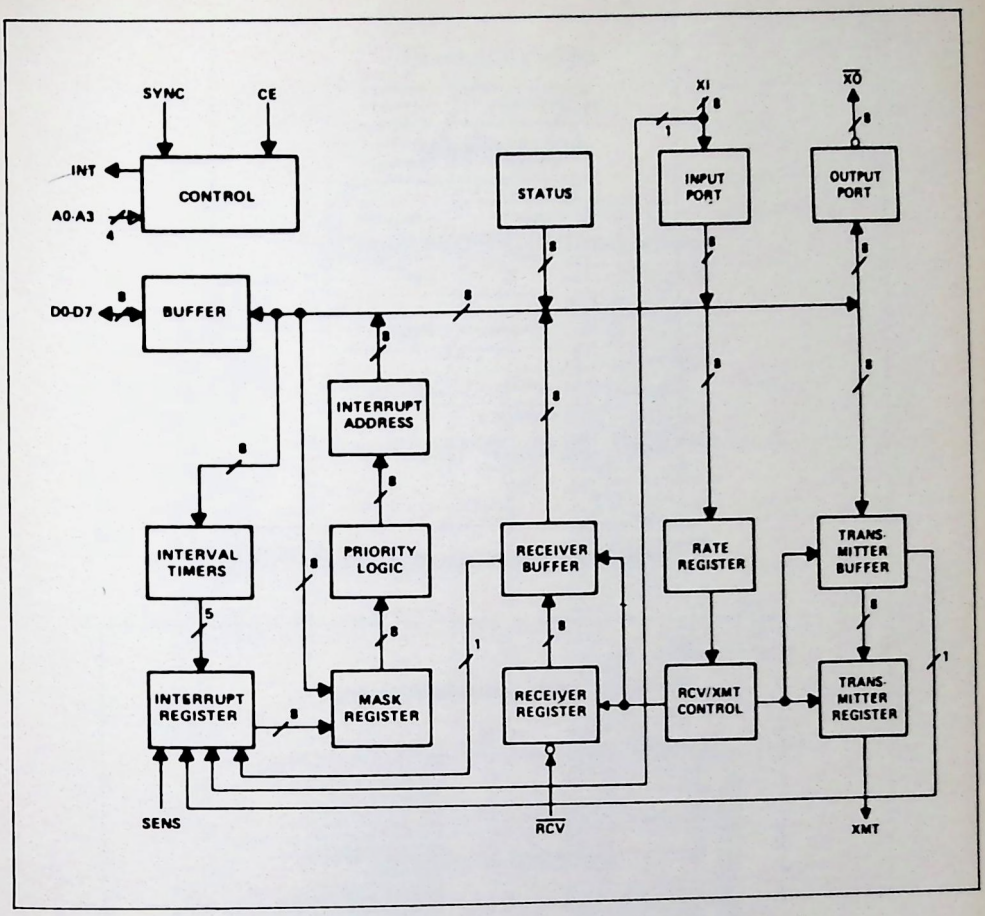
TMS 8080A/5501 8-Bit-Mikroprozessorsystem

Eigenschaften

- 8-Bit-Datenformat
- 8-, 16- und 24-Bit-Befehlsformat
- 64k-Byte-Programm-/Datenspeicher
- 7 Arbeitsregister
- 8 Interruptebenen
- 256 Ein- und 256 Ausgabe-Ports
- Multifunktion-Ein-/Ausgabe (UART, INTERRUPT, INTERVALLZEITGEBER und PARALLELE EIN-/AUSGABE)
- 78 Befehle
- 4 Adressierungsarten
- Stack-Architektur
- Befehlszykluszeit 5 μs typ



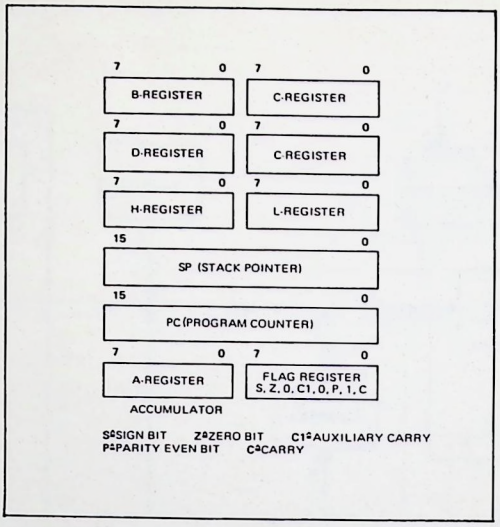
TMS 8080A/5501 8-Bit-Mikroprozessorsystem
Funktionsblockdiagramm TMS 8080A



TMS 8080A/5501 8-Bit-Mikroprozessorsystem
Funktionsblockdiagramm TMS 5501

CE = HIGH A3	A2	A1	A0	COMMAND	FUNCTION	TMS 8080 INSTRUCTION
L	L	L	L	READ RECEIVER BUFFER	RBn → Dn	MOV ra M
L	L	L	H	READ EXTERNAL INPUTS	XIn → Dn	MOV ra M
L	L	H	L	READ INTERRUPT ADDRESS	RST → Dn	MOV ra M
L	L	H	H	READ TMS 5501 STATUS	(STATUS) → Dn	MOV ra M
L	H	L	L	ISSUE DISCRETE COMMANDS	see below	MVI M b2
L	H	L	H	LOAD RATE REGISTER	see below	MVI M b2
L	H	H	L	LOAD TRANSMITTER BUFFER	Dn → TBn	MOV M ra
L	H	H	H	LOAD OUTPUT PORT	Dn → XOn	MOV M ra
H	L	L	L	LOAD MASK REGISTER	Dn → MRn	MVI M b2
H	L	L	H	LOAD INTERVAL TIMER 1	Dn → Timer 1	MVI M b2
H	L	H	L	LOAD INTERVAL TIMER 2	Dn → Timer 2	MVI M b2
H	L	H	H	LOAD INTERVAL TIMER 3	Dn → Timer 3	MVI M b2
H	H	L	L	LOAD INTERVAL TIMER 4	Dn → Timer 4	MVI M b2
H	H	L	H	LOAD INTERVAL TIMER 5	Dn → Timer 5	MVI M b2
H	H	H	L	NO FUNCTION	—	—
H	H	H	H	NO FUNCTION	—	—

TMS 5501-Befehle



Registerorganisation TMS 8080A

Akkumulator Befehlsgruppe

MNEMONIC		BYTE	STATUS AFFECTED	OPCODE D7-D4	OPCODE D3-D0	DISCRIPTION
ACI	b2	2	C,Z,S,P,C1	C	E	ADD IMMEDIATE TO A WITH CARRY
ADC	M	1	C,Z,S,P,C1	8	E	ADD MEMORY TO A WITH CARRY
ADC	r_a	1	C,Z,S,P,C1	8	8-F	ADD REGISTER r_a TO A WITH CARRY
ADD	M	1	C,Z,S,P,C1	8	6	ADD MEMORY TO A
ADD	r_a	1	C,Z,S,P,C1	8	0-7	ADD REGISTER r_a TO A
ADI	b2	2	C,Z,S,P,C1	C	6	ADD IMMEDIATE TO A
ANA	M	1	C,Z,S,P,C1	A	6	AND MEMORY WITH A
ANA	r_a	1	C,Z,S,P,C1	A	0-7	AND REGISTER r_a WITH A
ANI	b2	2	C,Z,S,P,C1	E	6	AND IMMEDIATE WITH A
CMA		1	— —	2	F	COMPLEMENT A
CMC		1	C	3	F	COMPLEMENT CARRY
CMP	M	1	C,Z,S,P,C1	B	E	COMPARE MEMORY WITH A
CMP	r_a	1	C,Z,S,P,C1	B	8-F	COMPARE REGISTER r_a WITH A
CPI	b2	2	C,Z,S,P,C1	F	E	COMPARE IMMEDIATE WITH A
DAA		1	C,Z,S,P,C1	2	7	DECIMAL ADJUST A
DAD	r_b	1	C	0-3	9	ADD REGISTER PAIR r_b WITH H&L
LDA	b3 b2	3	— —	3	A	LOAD A DIRECT
LDAX	r_c	1	— —	0,1	A	LOAD A INDIRECT WITH REGISTER PAIR r_c
ORA	M	1	C,Z,S,P,C1	B	6	OR MEMORY WITH A
ORA	r_a	1	C,Z,S,P,C1	B	0-7	OR REGISTER WITH A
ORI	b2	2	C,Z,S,P,C1	F	6	OR IMMEDIATE WITH A
RAL		1	C	1	7	ROTATE A LEFT THROUGH CARRY
RAR		1	C	1	F	ROTATE A RIGHT THROUGH CARRY
RLC		1	C	0	7	ROTATE A LEFT
RRC		1	C	0	F	ROTATE A RIGHT
SBB	M	1	C,Z,S,P,C1	9	E	SUBTRACT MEMORY FROM A WITH BORROW
SBB	r_a	1	C,Z,S,P,C1	9	8-F	SUBTRACT REGISTER r_a FROM A WITH BORROW
SBI	b2	2	C,Z,S,P,C1	D	E	SUBTRACT IMMEDIATE FROM A WITH BORROW
STA	b2 b3	3	— —	3	2	STORE A DIRECT
STAX	r_c	1	— —	0,1	2	STORE A INDIRECT WITH REGISTER PAIR r_c
STC		1	C	3	7	SET CARRY
SUB	M	1	C,Z,S,P,C1	9	6	SUBTRACT MEMORY FROM A
SUB	r_a	1	C,Z,S,P,C1	9	0-7	SUBTRACT REGISTER r_a FROM A
SUI	b2	2	C,Z,S,P,C1	D	6	SUBTRACT IMMEDIATE FROM A
XRA	M	1	C,Z,S,P,C1	A	E	EXCLUSIVE OR MEMORY WITH A
XRA	r_a	1	C,Z,S,P,C1	A	8-F	EXCLUSIVE OR REGISTER r_a WITH A
XRI	b2	2	C,Z,S,P,C1	E	E	EXCLUSIVE OR IMMEDIATE WITH A

Ein-/Ausgabebefehle

IN	b2	2	––	D	B	INPUT
OUT	b2	2	––	D	3	OUTPUT

Maschinenbefehle

HLT		1	––	7	6	HALT
NOP		1	––	0	0	NO OPERATION

Programmschrittzähler- und Stackbefehle

CALL	b3 b2	3	––	C	D	CALL UNCONDITIONAL
CC	b3 b2	3	––	D	C	CALL ON CARRY
CPE	b3 b2	3	––	E	C	CALL ON PARITY EVEN
CM	b3 b2	3	––	F	C	CALL ON MINUS
CZ	b3 b2	3	––	C	C	CALL ON ZERO
CNC	b3 b2	3	––	D	4	CALL ON NO CARRY
CPO	b3 b2	3	––	E	4	CALL ON PARITY ODD
CP	b3 b2	3	––	F	4	CALL ON POSITIVE
CNZ	b3 b2	3	––	C	4	CALL ON NO ZERO
DI		1	––	F	3	DISABLE INTERRUPTS
EI		1	––	F	B	ENABLE INTERRUPTS
JMP	b3 b2	3	––	C	3	JUMP UNCONDITIONAL
JC	b3 b2	3	––	D	A	JUMP ON CARRY
JP	b3 b2	3	––	F	2	JUMP ON POSITIVE
JPE	b3 b2	3	––	E	A	JUMP ON PARITY EVEN
JM	b3 b2	3	––	F	A	JUMP ON MINUS
JZ	b3 b2	3	––	C	A	JUMP ON ZERO
JNC	b3 b2	3	––	D	2	JUMP ON NO CARRY
JPO	b3 b2	3	––	E	2	JUMP ON PARITY ODD
JM	b3 b2	3	––	F	A	JUMP ON MINUS
JNZ	b3 b2	3	––	C	2	JUMP ON NO ZERO
PCHL		1	––	E	9	H&L TO PROGRAM COUNTER
POP	PSW	1	C,Z,S,P,C1	F	1	POP A AND FLAGS OFF STACK
POP	rd	1	––	C-E	1	POP REGISTER PAIR r_d OFF STACK
PUSH	PSW	1	––	F	5	PUSH A AND FLAGS ON STACK
PUSH	rd	1	––	C-E	5	PUSH REGISTER PAIR r_d ON STACK
RET		1	––	C	9	RETURN
RC		1	––	D	8	RETURN ON CARRY
RPE		1	––	E	8	RETURN ON PARITY EVEN
RM		1	––	F	8	RETURN ON MINUS
RZ		1	––	C	8	RETURN ON ZERO
RNC		1	––	D	0	RETURN ON NO CARRY
RPO		1	––	E	0	RETURN ON PARITY ODD
RP		1	––	F	0	RETURN ON POSITIVE
RNZ		1	––	C	0	RETURN ON NO ZERO
RST		1	––	C-F	7/F	RESTART
SPHL		1	––	F	9	H&L TO STACK POINTER

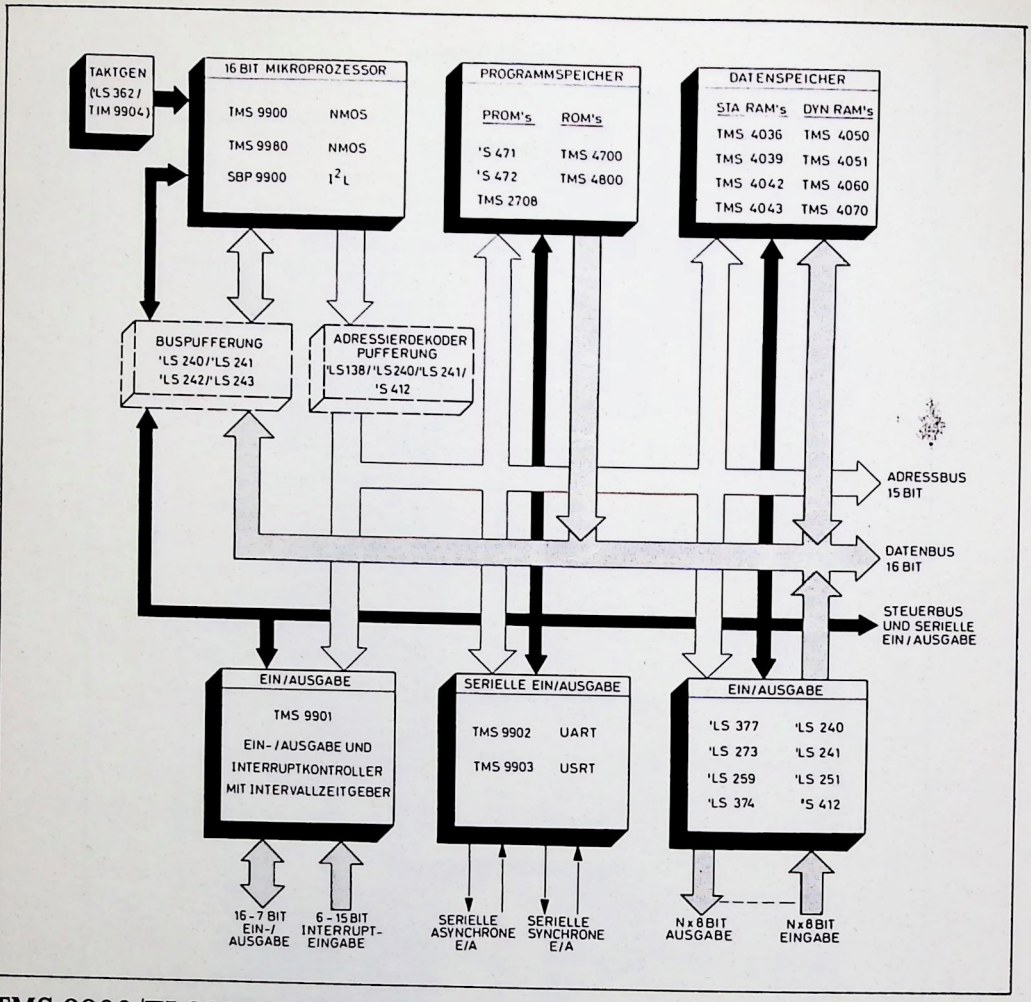
Register-Befehlsgruppe

DCR	M	1	Z,S,P,C1	3	5	DECREMENT MEMORY
DCR	ra	1	Z,S,P,C1	0-3	5/D	DECREMENT REGISTER r_a
DCX	rb	1	––	0-3	B	DECREMENT REGISTER PAIR r_b
INR	M	1	Z,S,P,C1	3	4	INCREMENT MEMORY
INR	ra	1	Z,S,P,C1	0-3	4/C	INCREMENT REGISTER r_a
INX	rb	1	––	0-3	3	INCREMENT REGISTER PAIR r_b
LHLD	b3 b2	3	––	2	A	LOAD H&L DIRECT
LXI	rb b3 b2	3	––	0-3	1	LOAD IMMEDIATE REGISTER PAIR r_b
MVI	M b2	2	––	3	6	MOVE IMMEDIATE MEMORY
MVI	ra b2	2	––	0-3	6/E	MOVE IMMEDIATE REGISTER r_a
MOV	M ra	1	––	7	0-7	MOVE REGISTER r_a TO MEMORY
MOV	ra M	1	––	4-7	6/E	MOVE MEMORY TO REGISTER r_a
MOV	ra1 ra2	1	––	4-7	0-F	MOVE REGISTER r_{a1} TO REGISTER r_{a2}
SHLD	b3 b2	3	––	2	2	STORE H&L REGISTER DIRECT
XCHG		1	––	E	B	EXCHANGE D&E, H&L REGISTERS
XTHL		1	––	E	3	EXCHANGE TOP OF STACK AND H&L

Befehlssymbole

SYMBOL	DISCRIPTION
ra	REGISTER B,C,D,E,H,L OR A
rb	REGISTER PAIR BC,DE,HL OR SP
rc	REGISTER PAIR BC OR DE
rd	REGISTER PAIR BC,DE OR HL
M	MEMORY ADDRESS DEFINED BY REGISTER H AND L
b2	SINGLE BYTE IMMEDIATE OPERAND
b3 b2	DOUBLE BYTE IMMEDIATE OPERAND

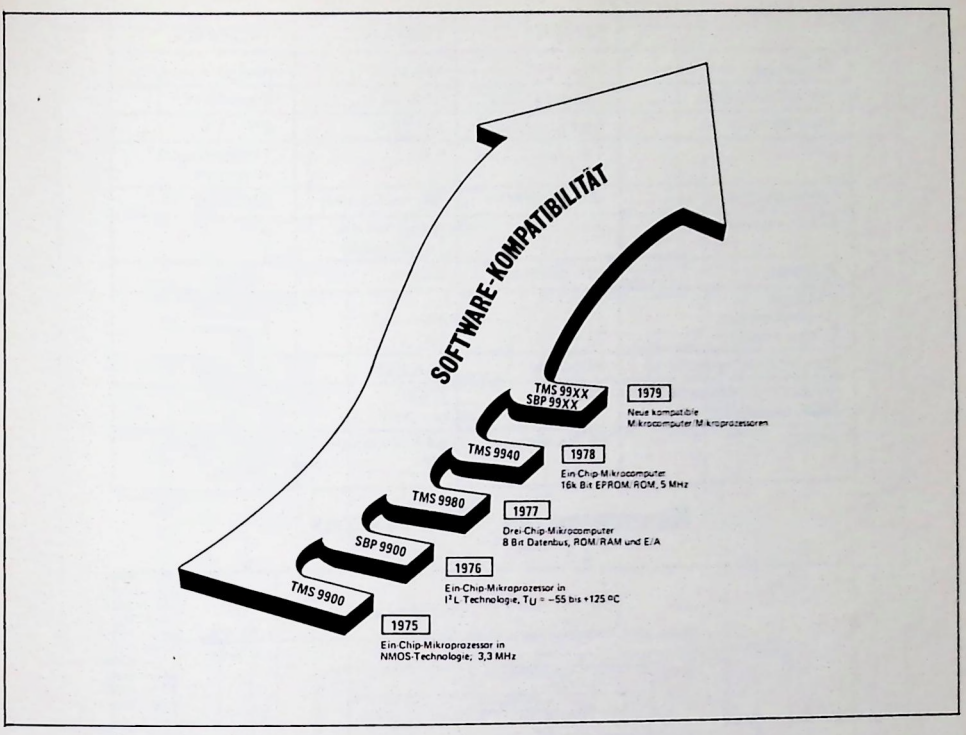
TMS 8080A/5501 8-Bit-Mikroprozessorsystem
Befehlsvorrat des TMS 8080A



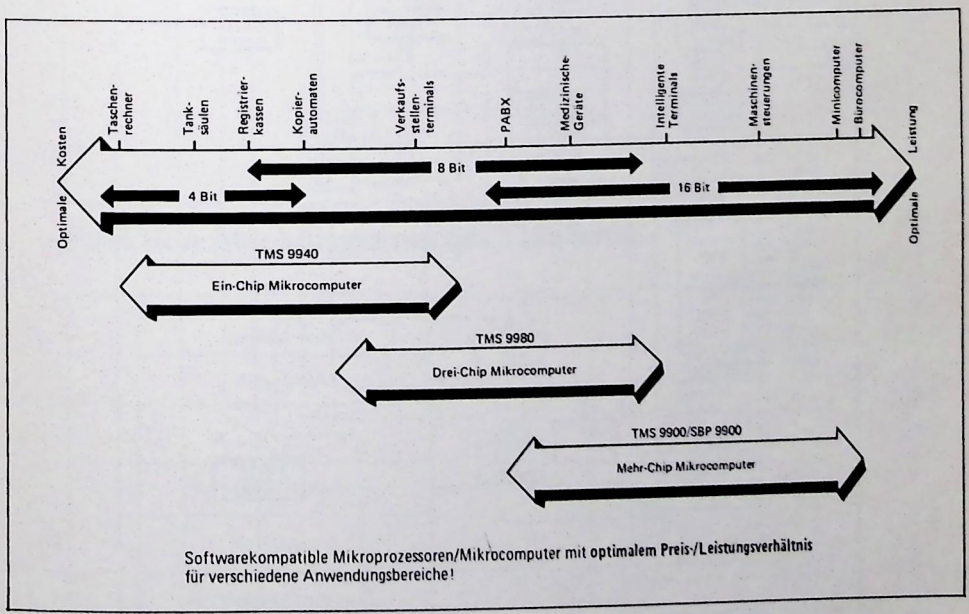
TMS 9900/TI 990 16-Bit-Mikro-/Minicomputer-Konzept

Eigenschaften

- 1-, 8- und 16-Bit Datenformat
- 16-, 32- und 48-Bit-Befehlsformat
- 32k-Worte (64k-Byte)-Programm-/ Datenspeicher
- 16 Arbeitsregister extern pro Programmteil
- 17 Interruptebenen
- Serielle Ein-/Ausgabe für langsame Peripherie
- 69 Befehle (inkl. Mult. und Div.)
- 7 Adressierungsarten
- Speicher-Speicher-Architektur
- Befehlszykluszeit 5 μs typ.
- Programmumschaltzeit 8 μs
- Volle TTL-Kompatibilität



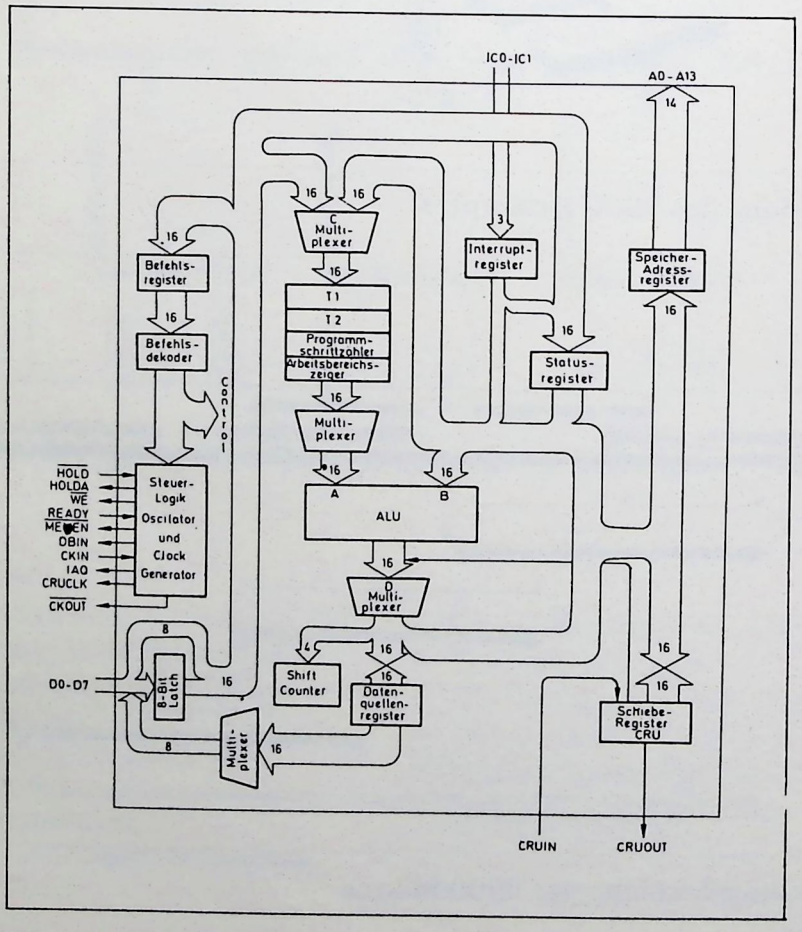
Entwicklung des 9900-Konzeptes



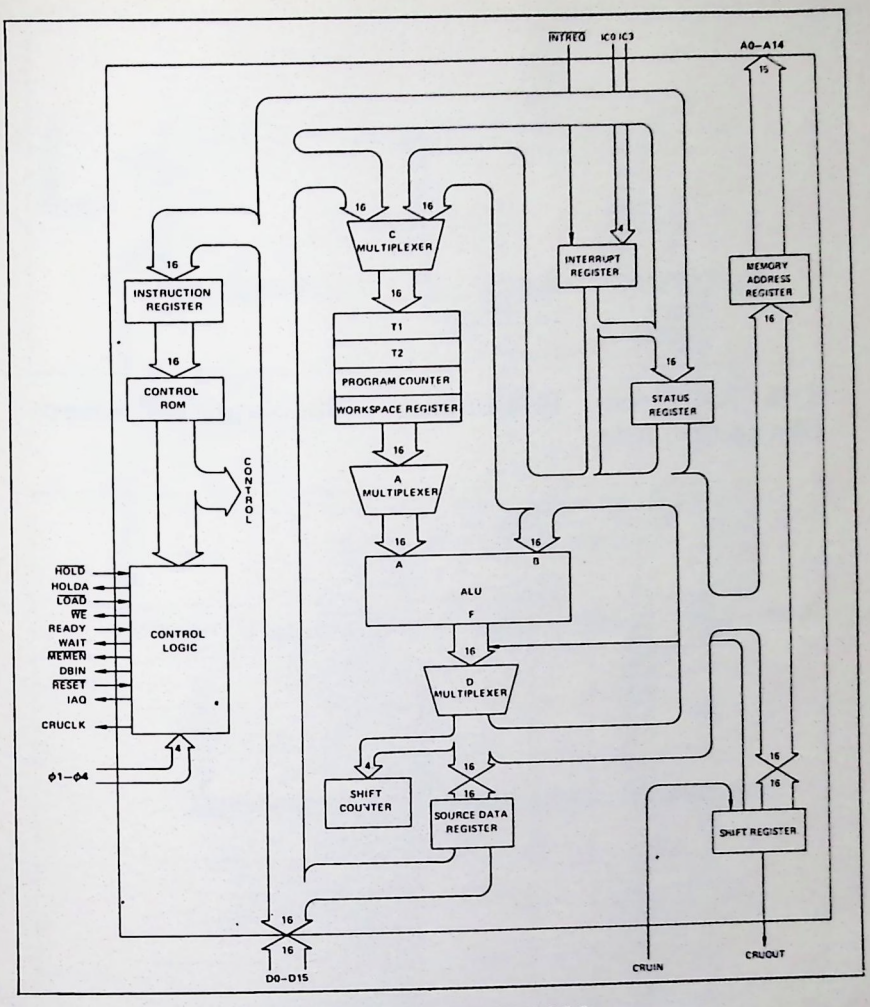
Anwendungsbereiche der 9900-Mikros

	SBP9900	TMS9900	TMS9980
Technologie	I^2L	NMOS	NMOS
Temperaturbereich	−55 bis +125 °C	0 bis 70 °C	0 bis 70 °C
Speisespannung	Eine Stromquelle	±5, 12 V	±5, 12 V
Takt	1 ϕ	4 ϕ	Taktgenerator integriert
Leistungsaufnahme typ.	500 mW/3 MHz	550 mW/3 MHz	990 mW/2 MHz
TTL-kompatibel	ja	ja (bis auf die Takteingabe)	ja
Ausgänge	Off. Kollektor	3 State	3 State
Interrupts vektorisierbar maskierbar	 17 15	 17 15	 6 4
Speicher direkt adressierbar	64K Byte	64K Byte	16K Byte
Anzahl der Ein/Ausgabe Bits, direkt adressierbar	je 4K	je 4K	je 2K
Gehäuse	64 Pin Keramik	64 Pin Keramik	40 Pin Keramik/ Plastik

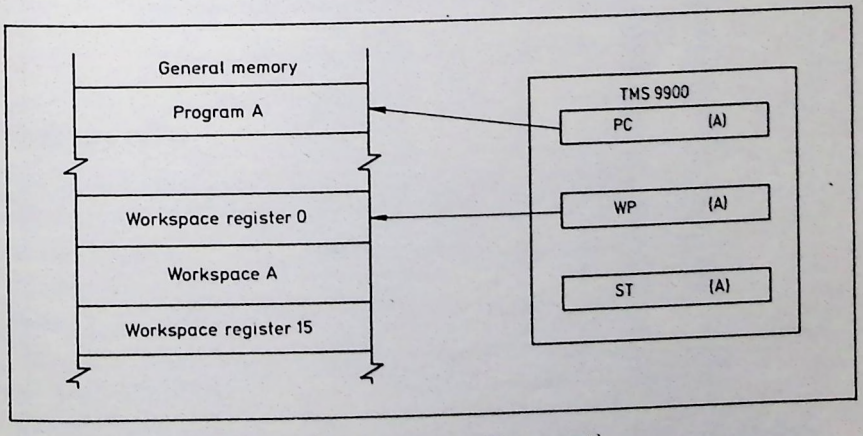
Kenndaten der 9900-Mikros



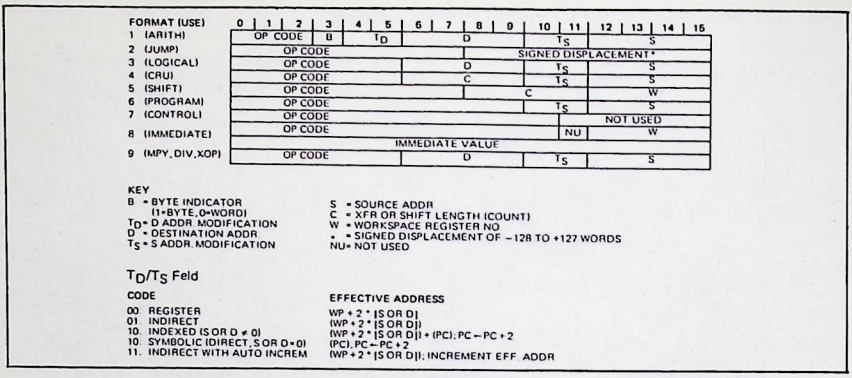
Funktionsblockdiagramm des TMS9980



Funktionsblockdiagramm des TMS 9900



Registerorganisation (Workspace-Konzept)



FORMAT (USE)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1 (ARITH)	OP CODE			B	T_D		D				T_S		S			
2 (JUMP)	OP CODE						SIGNED DISPLACEMENT*									
3 (LOGICAL)	OP CODE				D						T_S		S			
4 (CRU)	OP CODE				C						T_S		S			
5 (SHIFT)	OP CODE						C				W					
6 (PROGRAM)	OP CODE										T_S		S			
7 (CONTROL)	OP CODE										NOT USED					
8 (IMMEDIATE)	OP CODE										NU		W			
9 (MPY, DIV, XOP)	IMMEDIATE VALUE															
	OP CODE				D						T_S		S			

KEY

B = BYTE INDICATOR (1=BYTE, 0=WORD)
T_D = D ADDR. MODIFICATION
D = DESTINATION ADDR.
T_S = S ADDR. MODIFICATION
S = SOURCE ADDR.
C = XFR OR SHIFT LENGTH (COUNT)
W = WORKSPACE REGISTER NO.
* = SIGNED DISPLACEMENT OF −128 TO +127 WORDS
NU = NOT USED

T_D/T_S Feld

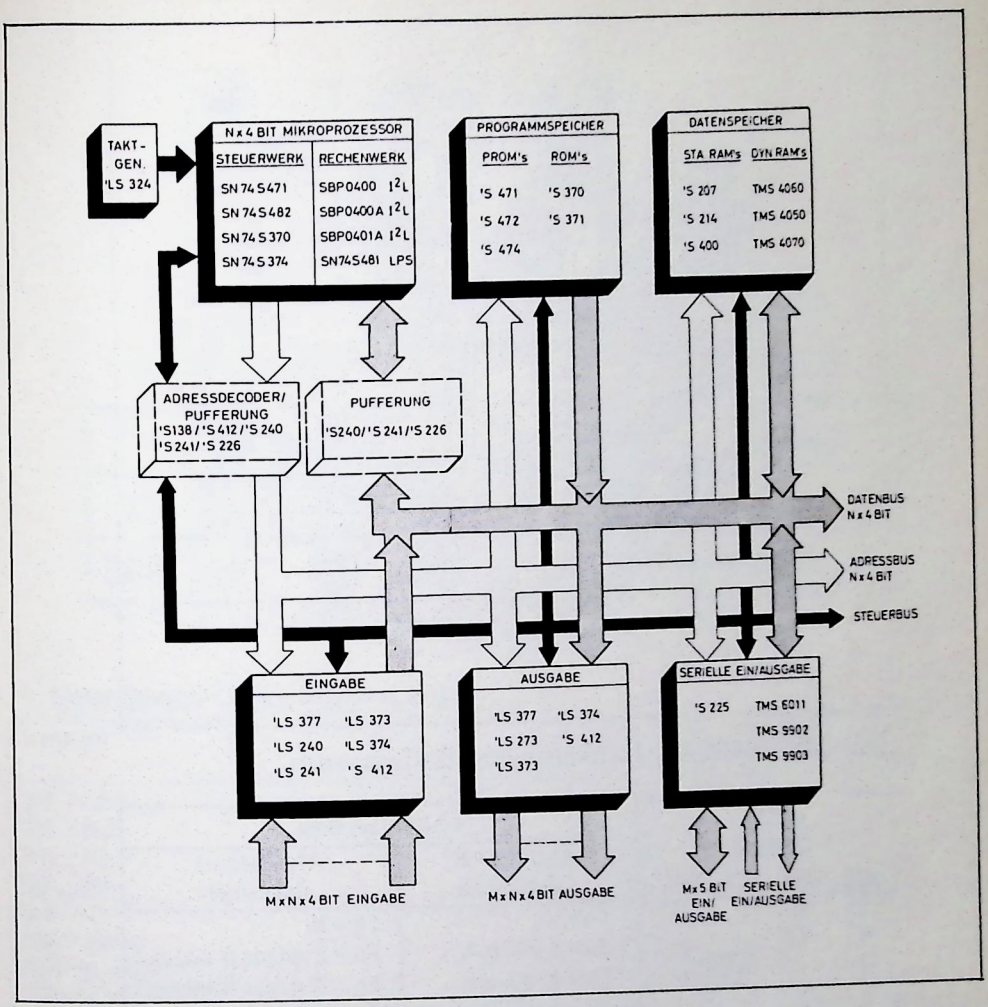
CODE	EFFECTIVE ADDRESS
00 REGISTER	WP + 2 * (S OR D)
01 INDIRECT	(WP + 2 * (S OR D))
10 INDEXED (S OR D ≠ 0)	(WP + 2 * (S OR D)) + (PC); PC ← PC + 2
10 SYMBOLIC (DIRECT, S OR D = 0)	(PC); PC ← PC + 2
11 INDIRECT WITH AUTO INCREM	(WP + 2 * (S OR D)); INCREMENT EFF. ADDR

TMS 9900/TI 990 16-Bit-Mikro-/Minicomputer-Konzept OP-Code Formate

MNEMONIC	OP CODE	FORMAT	RESULT COMPARED TO ZERO	STATUS AFFECTED	INSTRUCTIONS
A	A000	1	Y	0-4	ADD(WORD)
AB	B000	1	Y	0-5	ADD(BYTE)
ABS	0740	6	Y	0-4	ABSOLUTE VALUE
AI	0220	8	Y	0-4	ADD IMMEDIATE
ANDI	0240	8	Y	0-2	AND IMMEDIATE
B	0440	6	N	–	BRANCH
BL	0680	6	N	–	BRANCH AND LINK (W11)
BLWP	0400	6	N	–	BRANCH LOAD WORKSPACE POINTER
C	8000	1	N	0-2	COMPARE (WORD)
CB	9000	1	N	0-2,5	COMPARE (BYTE)
CI	0280	8	N	0-2	COMPARE IMMEDIATE
CKOF	03C0	7	N	–	EXTERNAL CONTROL
CKON	03A0	7	N	–	EXTERNAL CONTROL
CLR	04C0	6	N	–	CLEAR OPERAND
COC	2000	3	N	2	COMPARE ONES CORRESPONDING
CZC	2400	3	N	2	COMPARE ZEROES CORRESPONDING
DEC	0600	6	Y	0-4	DECREMENT (BY ONE)
DECT	0640	6	Y	0-4	DECREMENT (BY TWO)
DIV	3C00	9	N	4	DIVIDE
IDLE	0340	7	N	–	COMPUTER IDLE
INC	0580	6	Y	0-4	INCREMENT (BY ONE)
INCT	05C0	6	Y	0-4	INCREMENT (BY TWO)
INV	0540	6	Y	0-2	INVERT (ONES COMPLEMENT)
JEQ	1300	2	N	–	JUMP EQUAL (ST2=1)
JGT	1500	2	N	–	JUMP GREATER THAN (ST1=1)
JH	1B00	2	N	–	JUMP HIGH (ST0=1 AND ST2=0)
JHE	1400	2	N	–	JUMP HIGH OR EQUAL (ST0 or ST2=1)
JL	1A00	2	N	–	JUMP LOW (ST0 AND ST2=0)
JLE	1200	2	N	–	JUMP LOW OR EQUAL (ST0=0 OR ST2=1)
JLT	1100	2	N	–	JUMP LESS THAN (ST1 AND ST2=0)
JMP	1000	2	N	–	JUMP UNCONDITIONAL
JNC	1700	2	N	–	JUMP NO CARRY (ST3=0)
JNE	1600	2	N	–	JUMP NOT EQUAL (ST2=0)
JNO	1900	2	N	–	JUMP NO OVERFLOW (ST4=0)
JOC	1800	2	N	–	JUMP ON CARRY (ST3=1)
JOP	1C00	2	N	–	JUMP ODD PARITY (ST5=1)
LDCR	3000	4	Y	0-2,5	LOAD CRU
LI	0200	8	N	0-2	LOAD IMMEDIATE
LIMI	0300	8	N	12-15	LOAD IMMEDIATE TO INTERRUPT MASK
LREX	03E0	7	N	12-15	EXTERNAL CONTROL
LWPI	02E0	8	N	–	LOAD IMMEDIATE TO WORKSPACE POINTER
MOV	C000	1	Y	0-2	MOVE (WORD)
MOVB	D000	1	Y	0-2,5	MOVE (BYTE)
MPY	3800	9	N	–	MULTIPLY
NEG	0500	6	Y	0-4	NEGATE (TWO'S COMPLEMENT)
ORI	0260	8	Y	0-2	OR IMMEDIATE
RSET	0360	7	N	12-15	EXTERNAL CONTROL
RTWP	0380	7	N	0-6,12-15	RETURN WORKSPACE POINTER
S	6000	1	Y	0-4	SUBTRACT (WORD)
SB	7000	1	Y	0-5	SUBTRACT (BYTE)
SBO	1D00	2	N	–	SET CRU BIT TO ONE
SBZ	1E00	2	N	–	SET CRU BIT TO ZERO
SETO	0700	6	N	–	SET ONES
SLA	0A00	5	Y	0-4	SHIFT LEFT (ZERO FILL)
SOC	E000	1	Y	0-2	SET ONES CORRESPONDING (WORD)
SOCB	F000	1	Y	0-2,5	SET ONES CORRESPONDING (BYTE)
SRA	0800	5	Y	0-3	SHIFT RIGHT (MSB EXTENDED)
SRC	0B00	5	Y	0-3	SHIFT RIGHT CIRCULAR
SRL	0900	5	Y	0-3	SHIFT RIGHT (LEADING ZERO FILL)
STCR	3400	4	Y	0-2,5	STORE FROM CRU
STST	02C0	8	N	–	STORE STATUS REGISTER
STWP	02A0	8	N	–	STORE WORKSPACE POINTER
SWPB	06C0	6	N	–	SWAP BYTES
SZC	4000	1	Y	0-2	SET ZEROES CORRESPONDING (WORD)
SZCB	5000	1	Y	0-2,5	SET ZEROES CORRESPONDING (BYTE)
TB	1F00	2	N	2	TEST CRU BIT
X	0480	6	N	–	EXECUTE
XOP	2C00	9	N	6	EXTENDED OPERATION
XOR	2800	3	Y	0-2	EXCLUSIVE OR

ILLEGAL OP CODES 0000-01FF; 0320-033F; 0780-07FF; 0C00-0FFF

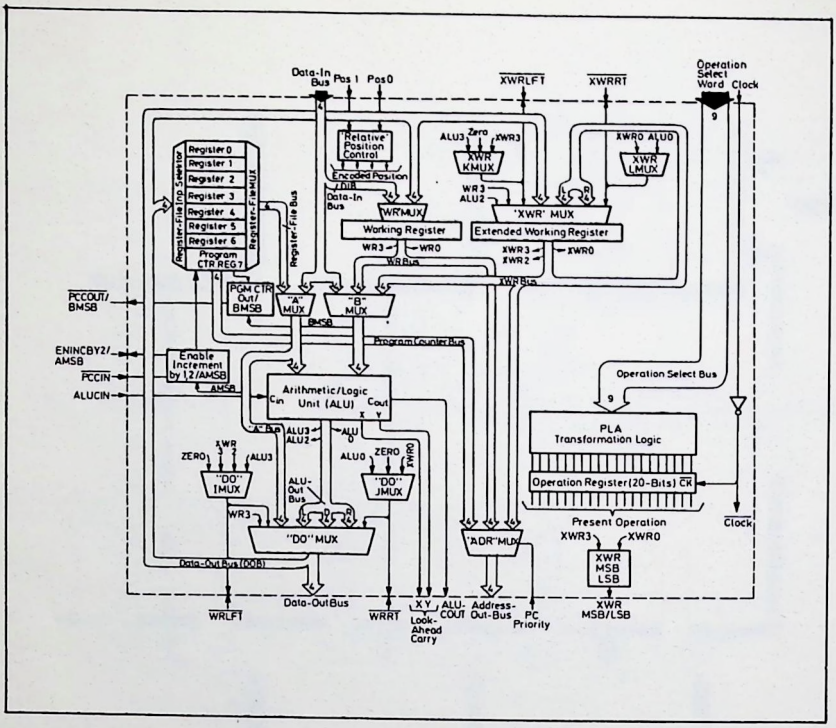
Befehlsvorrat des TMS 9980/9900, SBP 9900, TI 990/4



SBP 0400/0400A N-Bit Mikrocomputer Chip-Set

EIGENSCHAFTEN

- 4-Bit-Dateneingabebus 4-Bit-Datenausgabebus
- 4-Bit-Adressbus
- 7 Universalregister
- 2 Arbeitsregister
- Kaskadierb. 4-Bit-Programmschrittzähler
- 9-Bit-Mikrobefehlsformat
- 512 Mikrobefehle (459 sind nicht redundant)
- Voll kaskadierbar bis 64 Bit
- Statische Arbeitsweise
- Mikrobefehlszykluszeit 1 μs min.
- Volle TTL-Kompatibilität



SBP 0400/0400A Funktionsblockdiagramm

ALU OP-Field				Active-High Data	
OP3	OP2	OP1	OP0	ALUCIN = H (with carry)	ALUCIN = L (no carry)
L	L	L	L	Fn = L	Fn = H
L	L	L	H	Fn = B minus A	Fn = B minus A minus 1
L	L	H	L	Fn = A minus B	Fn = A minus B minus 1
L	L	H	H	Fn = A plus B plus 1	Fn = A plus B
L	H	L	L	Fn = B plus 1	Fn = B
L	H	L	H	Fn = $\overline{B}$ plus 1	Fn = $\overline{B}$
L	H	H	L	Fn = A plus 1	Fn = A
L	H	H	H	Fn = $\overline{A}$ plus 1	Fn = $\overline{A}$
H	L	L	L	Fn = AnBn	
H	L	L	H	Fn = $\overline{An \oplus Bn}$	
H	L	H	L	Fn = $\overline{An \oplus Bn}$	
H	L	H	H	Fn = $\overline{A}$nBn	
H	H	L	L	Fn = An$\overline{B}$n	
H	H	L	H	Fn = An + $\overline{B}$n	
H	H	H	L	Fn = $\overline{A}$n + Bn	
H	H	H	H	Fn = An + Bn	

ALU-Funktionstabelle

S-Field			Register
S2	S1	S0	Select
L	L	L	RF0
L	L	H	RF1
L	H	L	RF2
L	H	H	RF3
H	L	L	RF4
H	L	H	RF5
H	H	L	RF6
H	H	H	PC

Register-File Selection-Tabelle

OP-Field				D-Field		S-Field		
X	X	X	X	X	X	X	X	X
OP3	OP2	OP1	OP0	D1	D0	S2	S1	S0

Mikrobefehlsformat

1. Quellenoperand – Dateneingabebus (DIB)

Operation	OP3→OP0 OP-Field	D1 D0 D-Field	S2→S0 S-Field
DIB→DOB	HHHH	HH	LHL
DIB→DOB	LLLL	HH	LHL
DIB→RF	HHHH	HL	LLL→HHH
DIB→WR	LHHL	HL	XXX
DIB→WR	LHHL	HH	LHL
DIB→XWR	LLLH	HH	LHL
*DIB ALU WR→DOB	LLLL→HHHH	HH	LLL
*DIB ALU WR→WR	LLLL→HHHH	HH	LLH
DIB ALU WR→XWR	LLLL→HHHH	HH	HLL
DIB ALU XWR→DOB	LLLL→HHHH	HH	HHH
DIB ALU XWR→WR	LLLL→HHHH	HH	LHH
DIB ALU XWR→XWR	LLLL→HHHH	HH	HHL
(DIB plus WR plus ALUCIN) LCIR→WR, XWR	HLLH	HH	LHL
(DIB plus WR plus ALUCIN) RSA→WR, XWR	HLHH	HH	LHL
DIB plus RF plus ALUCIN→RF	LHHH	HL	LLL→HHH
DIB plus RF plus ALUCIN→WR	LHLL	HL	LLL→HHH
DIB plus RF plus ALUCIN→XWR	LHLH	HL	LLL→HHH
DIB plus WR plus ALUCIN→DOB	LHHH	HH	LHL
DIB plus WR plus ALUCIN→XWR	LLHH	HH	LHL
DIB plus XWR plus ALUCIN→WR	HHLL	HH	LHL
DIB plus XWR plus ALUCIN→XWR	HHLH	HH	LHL

* XWR→AOB

SBP 0400/0400A Befehlsvorrat

2. Quellenoperand – Erweitertes Arbeitsregister (XWR)

Operation	OP3 → OP0 OP-Field	D1 D0 D-Field	S2 → S0 S-Field
XWR ALU DIB → DOB	LLLL → HHHH	HH	HHH
XWR ALU DIB → WR	LLLL → HHHH	HH	LHH
XWR ALU DIB → XWR	LLLL → HHHH	HH	HHL
XWR plus ALUCIN → DOB	HHHL	HH	LHL
XWR plus ALUCIN → RF	HHHL	HL	LLL → HHH
XWR plus DIB plus ALUCIN → WR	HHLL	HH	LHL
XWR plus DIB plus ALUCIN → XWR	HHLH	HH	LHL
XWR plus RF plus ALUCIN → WR	HHLL	HL	LLL → HHH
XWR plus RF plus ALUCIN → XWR	HHLH	HL	LLL → HHH

3. Quellenoperand – Register File (RF)

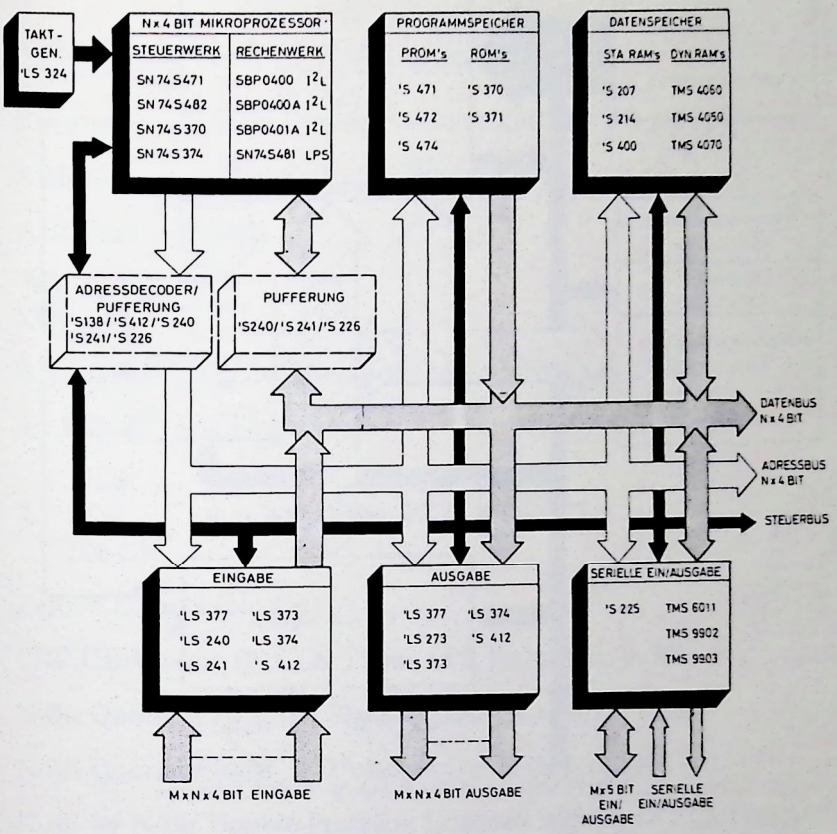
Operation	OP3 → OP0 OP-Field	D1 D0 D-Field	S2 → S0 S-Field
RF → DOB	LLLL	HL	LLL → HHH
RF → XWR	LLLH	HL	LLL → HHH
RF ALU WR → RF	LLLL → HHHH	LL	LLL → HHH
RF ALU WR → WR	LLLL → HHHH	LH	LLL → HHH
RF plus DIB plus ALUCIN → RF	LHHH	HL	LLL → HHH
RF plus DIB plus ALUCIN → WR	LHLL	HL	LLL → HHH
RF plus DIB plus ALUCIN → XWR	LHLH	HL	LLL → HHH
(RF plus WR plus ALUCIN) LCIR → WR, XWR	HLLH	HL	LLL → HHH
(RF plus WR plus ALUCIN) RSA → WR, XWR	HLHH	HL	LLL → HHH
RF plus WR plus ALUCIN → XWR	LLHH	HL	LLL → HHH
RF plus XWR plus ALUCIN → WR	HHLL	HL	LLL → HHH
RF plus XWR plus ALUCIN → XWR	HHLH	HL	LLL → HHH

4. Quellenoperand – Arbeitsregister (WR)

Operation	OP3 → OP0 OP-Field	D1 D0 D-Field	S2 → S0 S-Field
*WR ALU DIB → DOB	LLLL HHHH	HH	LLL
*WR ALU DIB → WR	LLLL HHHH	HH	LLH
WR ALU DIB → XWR	LLLL HHHH	HH	HLL
WR ALU RF → RF	LLLL HHHH	LL	LLL → HHH
WR ALU RF → WR	LLLL HHHH	LH	LLL → HHH
(WR minus DIB minus 1 plus ALUCIN) LCIR → WR, XWR	HLLL	HH	LHL
(WR minus DIB minus 1 plus ALUCIN) RSA → WR, XWR	LLHL	HH	LHL
(WR minus RF minus 1 plus ALUCIN) LCIR → WR, XWR	HLLL	HL	LLL → HHH
(WR minus RF minus 1 plus ALUCIN) RSA → WR, XWR	LLHL	HL	LLL → HHH
(WR plus ALUCIN) RSA → WR, XWR	HLHL	HL	XXX
(WR plus ALUCIN) RSA → WR, XWR	HLHL	HH	LHL
(WR plus ALUCIN) LCIR → WR	LLHH	HH	HLH
(WR plus ALUCIN) LCIR → WR	HLHH	HH	HLH
(WR plus ALUCIN) LSA → WR	LLHL	HH	HLH
(WR plus ALUCIN) LSL → WR	HLHL	HH	HLH
(WR plus ALUCIN) RCIR → WR	LLLH	HH	HLH
(WR plus ALUCIN) RCIR → WR	HLLH	HH	HLH
(WR plus ALUCIN) RSA → WR	LLLL	HH	HLH
(WR plus ALUCIN) RSL → WR	HLLL	HH	HLH
(WR plus ALUCIN, XWR) LCIR → (WR, XWR)	HHHH	HH	HLH
(WR plus ALUCIN, XWR) LCIR → (WR, XWR)	LHHH	HH	HLH
(WR plus ALUCIN, XWR) LSA → (WR, XWR)	LHHL	HH	HLH
(WR plus ALUCIN, XWR) LSL → (WR, XWR)	HHHL	HH	HLH
(WR plus ALUCIN, XWR) RCIR → (WR, XWR)	HHLH	HH	HLH
(WR plus ALUCIN, XWR) RCIR → (WR, XWR)	LHLH	HH	HLH
(WR plus ALUCIN, XWR) RSA → (WR, XWR)	LHLL	HH	HLH
(WR plus ALUCIN, XWR) RSL → (WR, XWR)	HHLL	HH	HLH
WR plus DIB plus ALUCIN → DOB	LHHH	HH	LHL
WR plus DIB plus ALUCIN → XWR	LLHH	HH	LHL
(WR plus DIB plus ALUCIN) LCIR → WR, XWR	HLLH	HH	LHL
(WR plus DIB plus ALUCIN) RSA → WR, XWR	HLHH	HH	LHL
WR plus RF plus ALUCIN → XWR	LLHH	HL	LLL → HHH
(WR plus RF plus ALUCIN) LCIR → WR, XWR	HLLH	HL	LLL → HHH
(WR plus RF plus ALUCIN) RSA → WR, XWR	HLHH	HL	LLL → HHH

* XWR → AOB

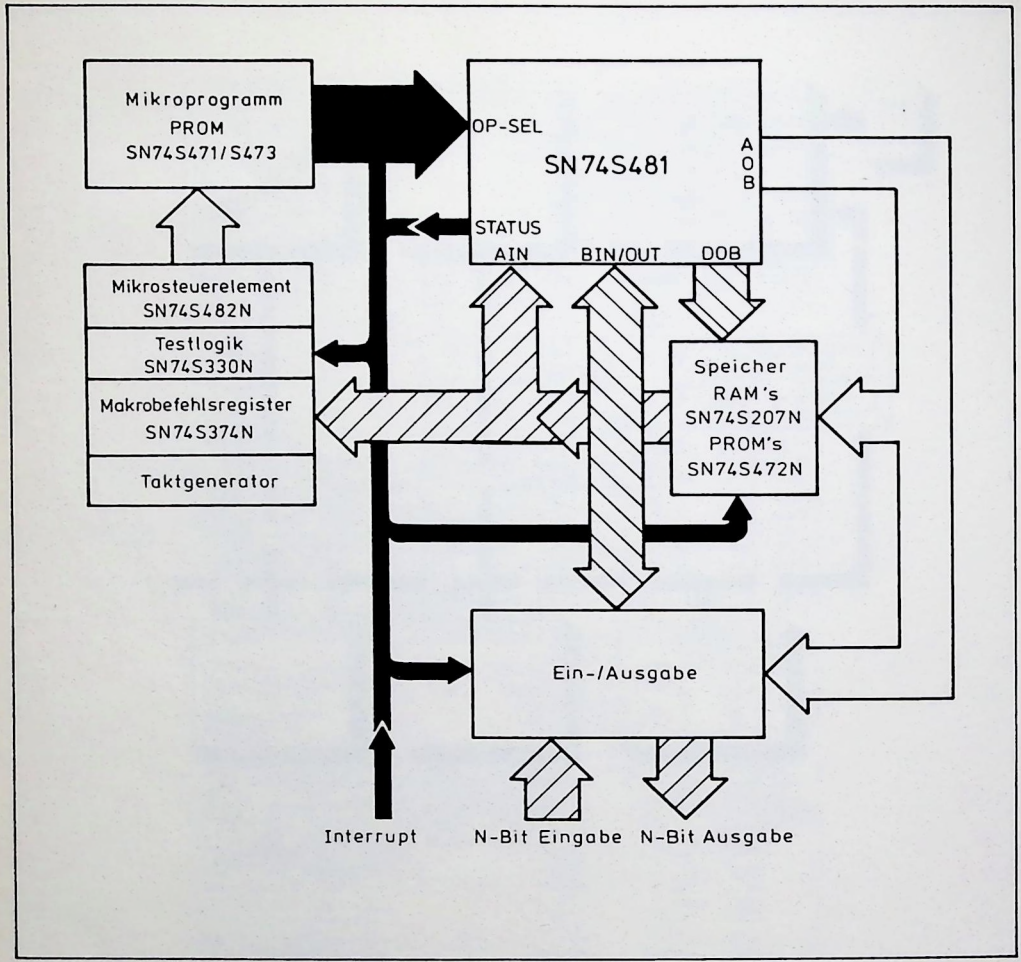
SBP 0400/0400A Befehlsvorrat



SN 74S481/S482 N-Bit Mikrocomputer Chip-Set

Eigenschaften

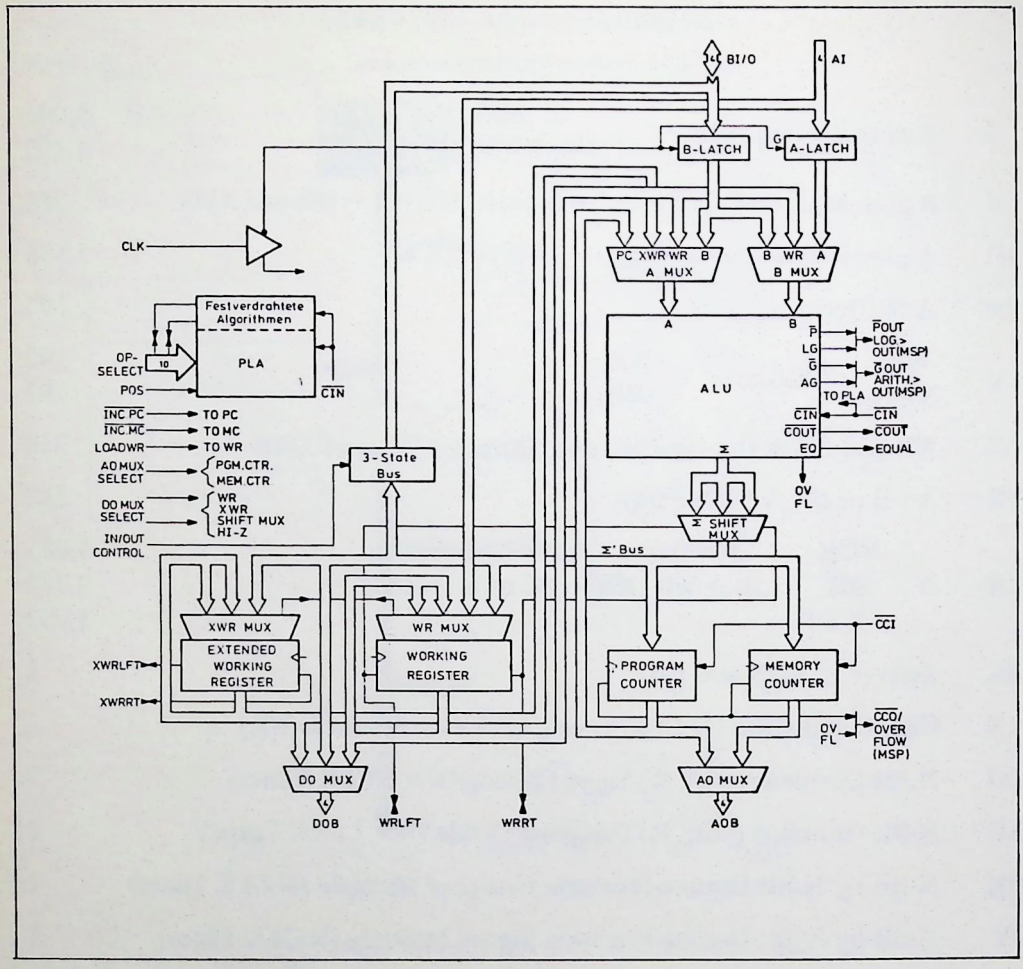
- Bidirektionaler 4-Bit-Ein-/Ausgabe-Datenbus
- 2 kaskadierbare 4-Bit-Zähler
- 10-Bit-Mikrobefehlsformat
- 24780 Mikrobefehle
- 5 Makrobefehle (z.B. Mult., Div.)
- Mikrobefehlszykluszeit 100 ns typ.



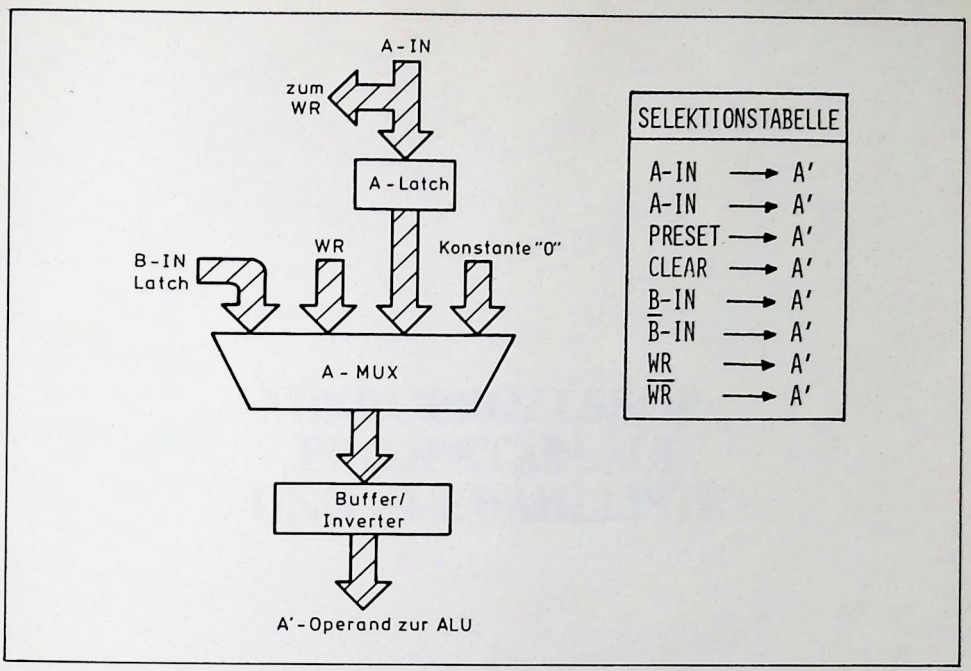
SN74S481/482 N-Bit Mikrocomputer Chip-Set

Befehlsübersicht SN74S481

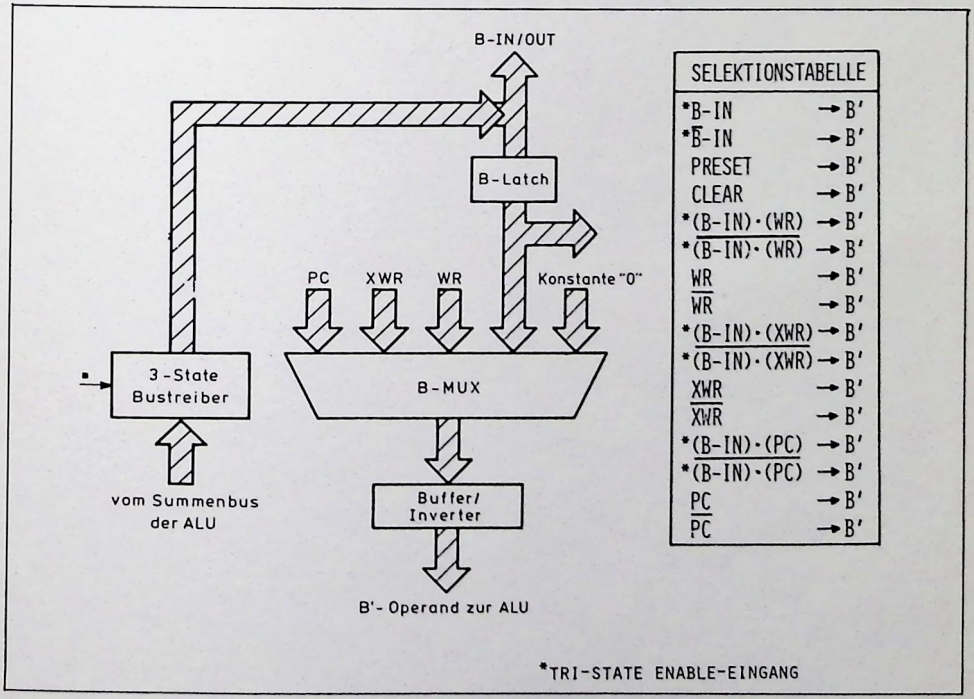
Form	Mikro/Makro-Operationsformen		No. of OP's
I	A plus B plus Alucin →	Σ' BUS ONLY	4,096
		WR, XWR, PC or MC	8,192
II	B plus A plus Alucin Double-Precision Shifted → WR and XWR		384
III	A plus B plus Alucin Shifted → WR or XWR		1,536
IV	A Shifted → Σ' BUS		192
V	WR	Shifted → WR	192
	XWR	XWR	192
VI	WR and XWR Double-Precision Shifted → WR and XWR		256
VII	A : B or B : A (Compare)		512
VIII	A NOR	B → WR, XWR, PC or B IN/OUT	3,072
	A OR	B → WR, XWR, PC or B IN/OUT	3,072
	A EX-OR	B → WR, XWR, PC or B IN/OUT	3,072
IX	Zero → Σ' BUS (No Op)		1
X	CRC Calculation (N CLK Times, CPR Partial Sum in WR)		1
XI	N-Bit Quotient (wirh R) Signed Divide (N + 3 CLK Times)		5
XII	N-Bit Quotient (with R) Unsigned Divide (N + 1 CLK Times)		3
XIII	N-Bit by N-Bit Double-Precision Unsigned Multiply (N CLK Times)		1
XIV	N-Bit by N-Bit Double-Precision Signed Multiply (N CLK Times)		1
		Summe der Operationen	24,780



SN74S481/482 Architektur

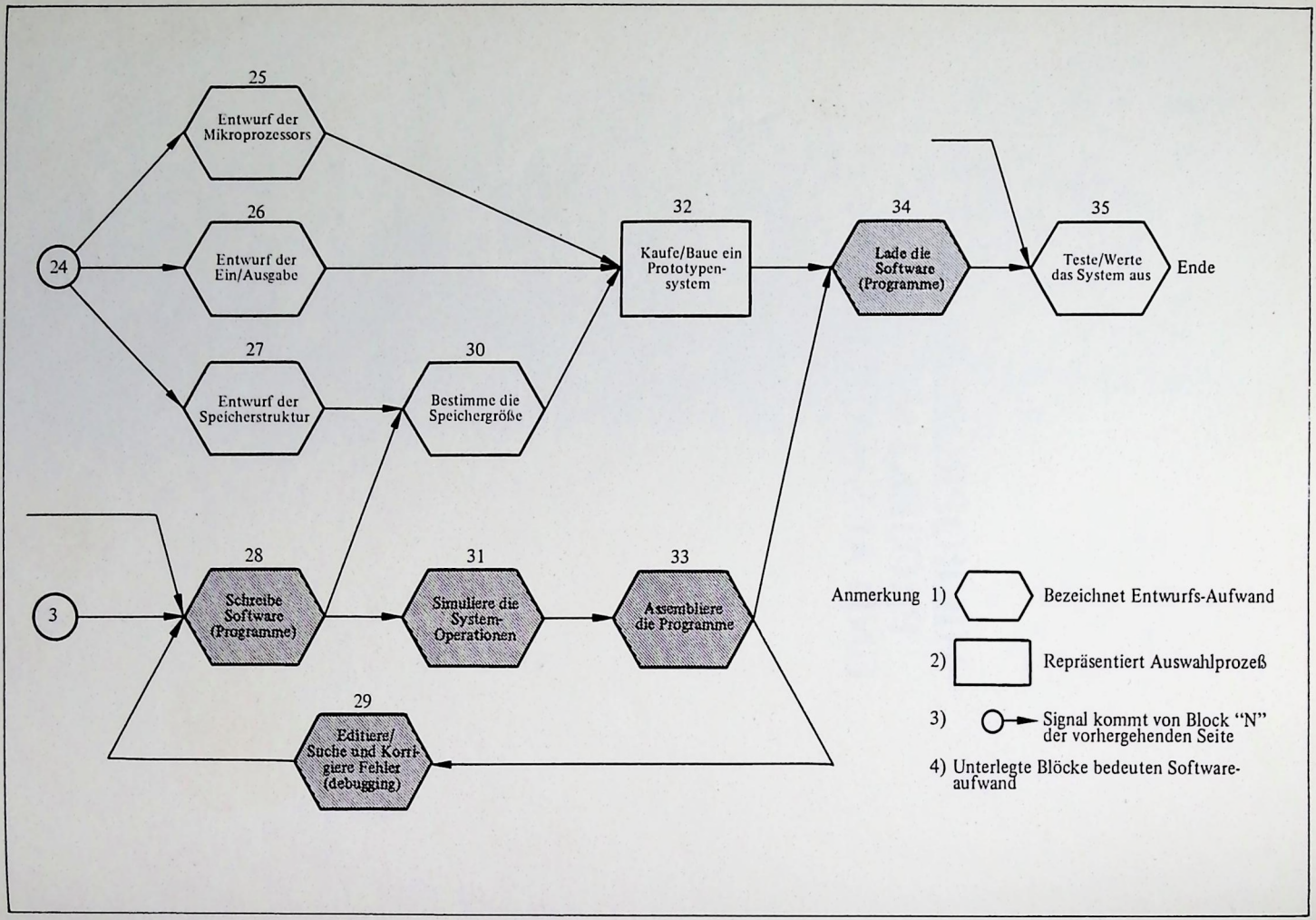


SN74S481/482 Selektion des Operanden A

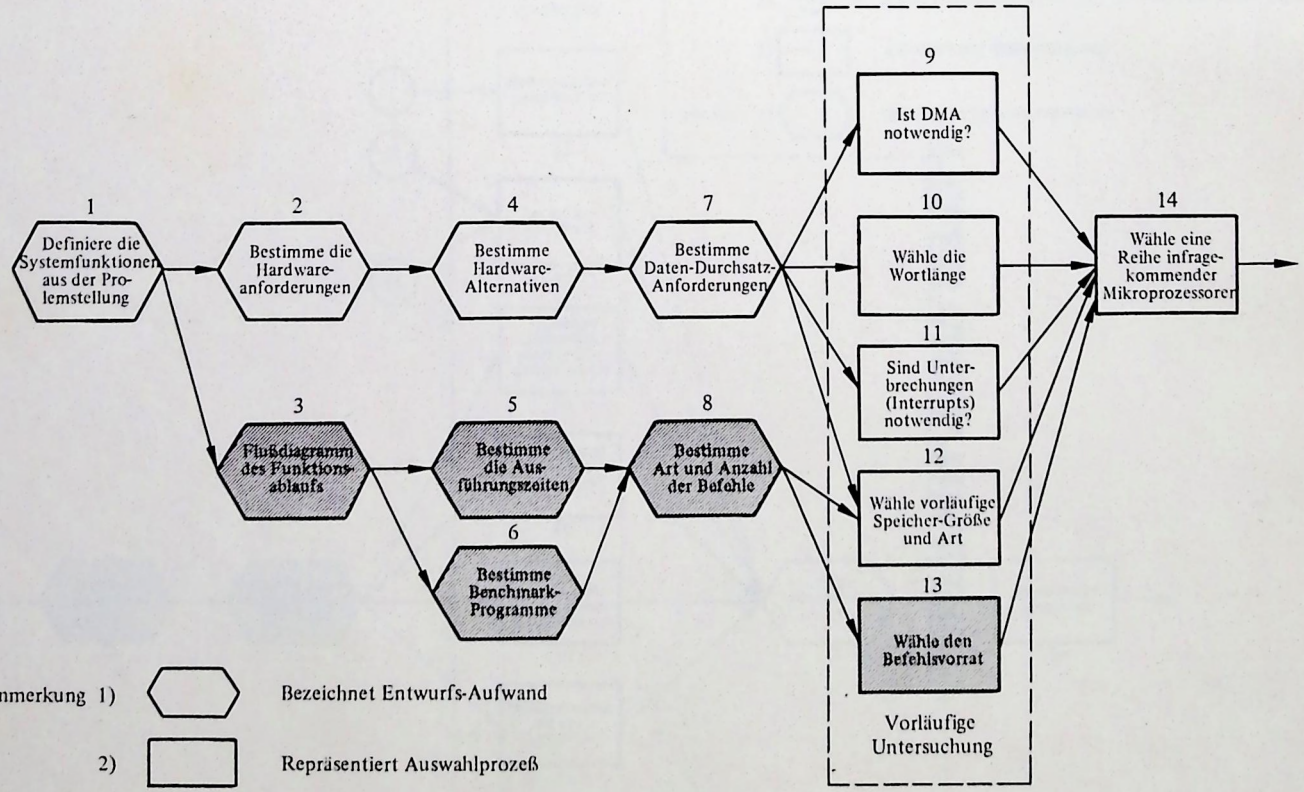


SN74S481/482 Selektion des Operanden B

MIKROPROZESSOR-PROJEKTABLAUF UND AUSWAHLLISTE


25
Entwurf der Mikroprozessors
26
Entwurf der Ein/Ausgabe
27
Entwurf der Speicherstruktur
24
30
Bestimme die Speichergröße
32
Kaufe/Baue ein Prototypen-system
34
Lade die Software (Programme)
35
Teste/Werte das System aus
Ende
3
28
Schreibe Software (Programme)
31
Simuliere die System-Operationen
33
Assembliere die Programme
29
Editiere/ Suche und Korrigiere Fehler (debugging)
Anmerkung 1) Bezeichnet Entwurfs-Aufwand
2) Repräsentiert Auswahlprozeß
3) Signal kommt von Block "N" der vorhergehenden Seite
4) Unterlegte Blöcke bedeuten Software-aufwand

Flußdiagramm für den Mikroprozessor-Entwurf / Richtlinien für die Auswahl



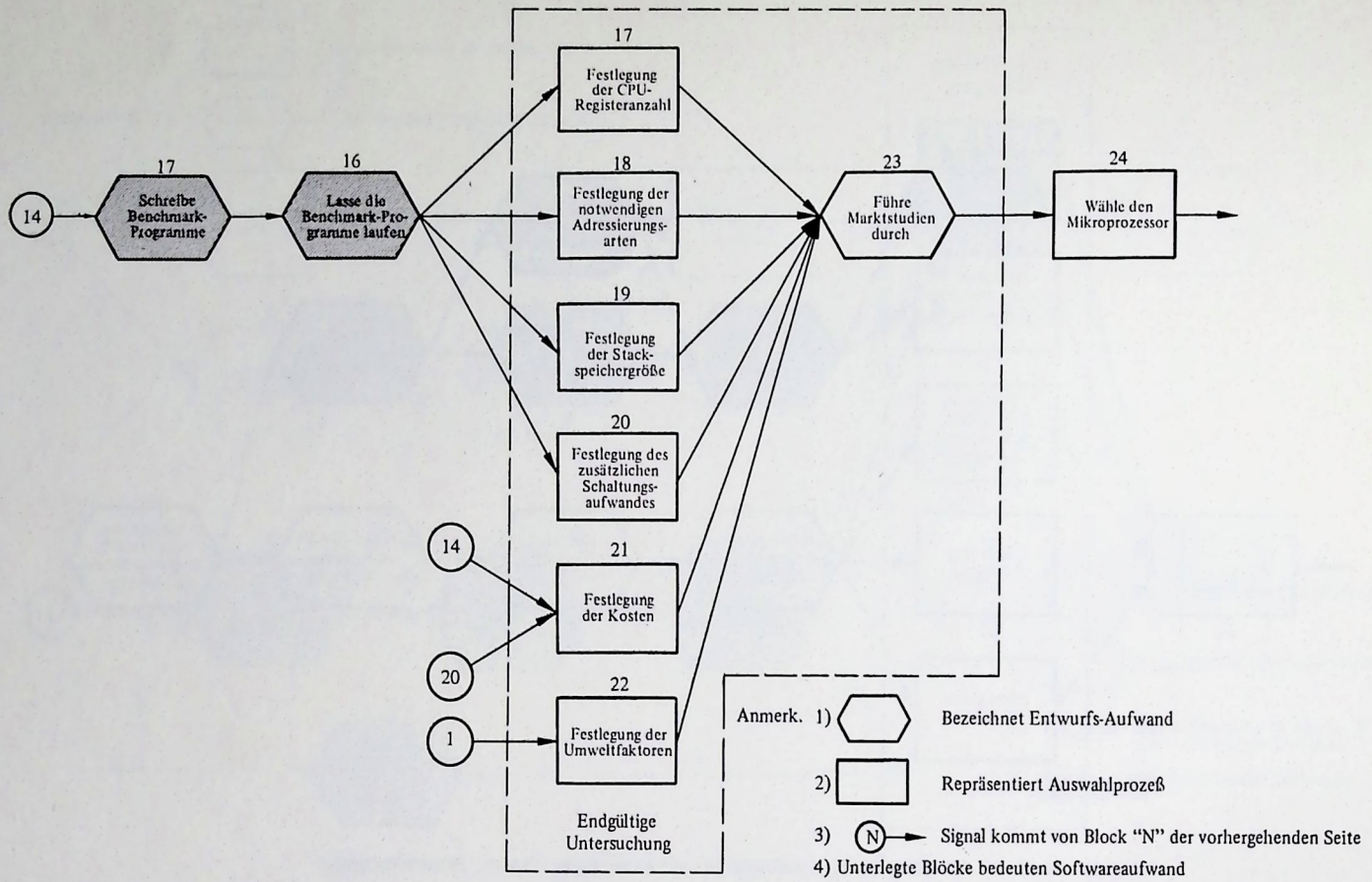
Anmerkung 1) Bezeichnet Entwurfs-Aufwand

2) Repräsentiert Auswahlprozeß

3) (N) → Signal kommt von Block "N" der vorhergehenden Seite

4) Unterlegte Blöcke bedeuten Softwareaufwand

Flußdiagramm für den Mikroprozessor-Entwurf / Richtlinien für die Auswahl (Fortsetzung)



LITERATURHINWEISE

Deutsche Literaturhinweise

- **Marktübersichten**

[1] Gößler, R., Schwerte, J., "Auf dem Weg zur Mikroprozessor-Praxis" *Elektronik* 1976, H.3.

[2] Schneider, P., "Die Qual der Wahl beim Mikroprozessor" *Der Elektroniker* 1975, Nr. 9, Teil I + II.

[3] Timm, V., "Im Blickpunkt: ROM's, PROM's und PLA's *Elektronik* 1976, H.5.

[4] Dipl.-Ing. Manfred Schlenker, "Im Blickpunkt: Bipolare und MOS-Schreib-/Lesespeicher (RAMs)" *Elektronik* 1976, H.10, S. 57-68.

[5] Dipl.-Ing. Reinhard Gößler, "Ein-/Ausgabe-Bausteine für Mikroprozessoren, Einführung – Übersicht – Zusammenstellung" *Elektronik* 1976, H.11.

- **Grundlagen zur weiteren Einarbeitung**

[6] Blomeyer-Bartenstein, H.P., "Mikroprozessoren und Mikrocomputer" *Siemens AG*, Bestell-Nr. B 12/1420.

[7] "Mikrocomputer", AT-Fachverlag GmbH, Stuttgart. Sonderdruck aus der Fachzeitschrift *"Der Elektroniker"*, 1975.

[8] "Mikroprozessoren", Franzis-Verlag GmbH, München. Sonderdruck aus der Fachzeitschrift *"Elektronik"*, 11/1976.

[9] Bast, M., "I^2L Neue bipolare Logik mit hoher Packungsdichte" *Elektronik Praxis*, 1975, H.10.

[10] Huse, H., "I^2L: Die Technologie und ihre Anwendung in einem schnellen 4-Bit-Mikroprozessorelement" *Elektronik* 1976, H.2

[11] "Mikro oder Mini" *Elektronik Praxis* 1975, H.12.

[12] Neunert, H., "Systemtechnik mit Mikrocomputern" *Elektronik* 1974, H.10, S. 391-395.

[13] Habgood, D., "Wie wär's mit einem Mikroprozessor? " *Elektronik* 1974, H.10, S. 379-382.

[14] Fertl, W., "MPS 10 – Ein modularer Mikrocomputer" *Elektronik* 1974, H.10, S. 383-386.

[15] Münchrath, R., "Tendenzen der Minicomputer-Entwicklung" *Elektronik* 1973, H.10, S. 345-348.

[16] Scheytt, W., "Minicomputer" *Elektronik* 1975, H.10, S. 82-90.

[17] Dieckmann, H.W., "Mikroprozessor. Computer statt festverdrahteter Logik" *Funkschau* 1975, H.26 S. 53-56.

[18] Brümmer, Hans, "Einführung in die Zuverlässigkeit elektronischer Bauelemente, Geräte und Systeme" *Der Elektroniker* 1976, H.2, EL 1-11.

Englische Literaturhinweise

[1] Altman, L., "Logic's Leap Ahead Creates New Design Tools for Old and New Applications," *Electronics* (February 21, 1974).

[2] Altman, L., "Single-Chip Microprocessors Open Up a New World of Applications," *Electronics*, Vol. 47, No. 8 (April 18, 1974), pp. 81-87.

[3] Bailey, S.J., "Microprocessor: Candidate for Distributed Computing Control," *Control Engineering*, Vol. 21, No. 3 (March 1974), pp. 40-44.

[4] Bell, C.G. and Kaman, C., "Microprocessor – Another Member of the Mini(Mal) Computer Family," *IEEE International Convention Technical Papers* (1974), Paper No. 24/4.

[5] Boehme, C.R., "Use of a Microprocessor as a Peripheral System Controller," *8th IEEE Computer Society International Conference Digest of Papers* (1974). "Computer Peripherals – Benefactor of Bottleneck? " pp. 181-184.

[6] Chiang, A.C.L., McCaskill, R. "Two New Approaches Simplify Testing of Microprocessors," *Electronics* (July 22, 1976), pp. 100-105.

[7] Cropper, L.C. and Whiting, J., "Evaluation of the Use of Microprocessors in CRT Display Terminals," *8th IEEE Computer Society International Conference Digest of Papers* (1974), "Computer Peripherals – Benefactor or Bottleneck? " pp. 185-187.

[8] Cropper, L.C. and Whiting, J.W., "Microprocessors in CRT Terminals," *Computer* (August 1974).

[9] Cushman, R.H., "Don't Overlook the 4-Bit Microprocessor: They're Here and They're Cheap," *EDN*, Vol. 19, No. 4 (February 20, 1974), pp. 44-50.

[10] Cushman, R.H., "Intel 8080: The First of the Second-Generation Microprocessors," *EDN*, Vol. 19, No. 9 (May 5, 1974), pp. 3036.

[11] Cushman, R.H., "Microprocessors are Changing Your Future. Are You Prepared? " *EDN*, Vol. 18, No. 21 (November 5, 1973), pp. 26-32.

[12] Cushman, R.H., "Microprocessors Are Rapidly Gaining on Minicomputers," *EDN*, Vol. 19, No. 10 (May 20, 1974), pp. 16-19.

[13] Cushman, R.H., "Understand the 8-Bit Microprocessor: You'll See A Lot of It," *EDN*, Vol. 19, No. 2 (January 20, 1974), pp. 48-54.

[14] Cushman, R.H., "Understanding the Microprocessor is no Trivial Task," *EDN*, Vol. 18, No. 22 (November 20, 1973), pp. 42-49.

[15] Cushman, R.H., "What Can You Do With a Microprocessor? " *EDN*, Vol. 19, No. 6 (March 20, 1974), pp. 42-48.

[16] Davis, S., "Fresh View of Mini- and Microcomputers," *Computer Design*, Vol. 13, No. 5 (May 1974), pp. 67-79.

[17] Electrical Research Association, Computers and Automation Division, *Microprocessor: An ERA Assessment of LSI Computer Components*, Published by Ovum, Ltd., 22 Grays Inn Road, London WC1 (1973).

[18] Eloe, E.E. and Scott, R.T., Jr., "Helicopter Flight Performance System Using an LSI Microprocessor," AD 765680/4 (June 1973), 59 p., Available from National Technical Information Service, Springfield, Virginia 22151.

[19] Falk, H., "Microcomputer Software Makes Its Debut," *IEEE Spectrum* (October 1974).

[20] Fisher, E. and others, "Stores Stock Microcomputer System: The MCS-8 8-Bit Processor Controller," UCID-16341 (July 10, 1973), 15 p., Available from National Technical Information Service, Springfield, Virginia 22151.

[21] Foster, C.C., "Something New – The INTEL MCS-4 Microcomputer Set," *Computer Architecture News*, Vol. 1, No. 2 (April 1972), pp. 16-17.

[22] Garrow, B., Johnson, J., Maen, M., "The 'Super Component': The One Board Computer with Programable I/O," *Electronics* (February 1976), pp. 77-84.

[23] Gladstone, B., "Designing with Microprocessors Instead of Wired Logic Asks More of Designers," *Electronics*, Vol. 46, No. 21 (October 11, 1973), pp. 91-104.

[24] Gott, W., "Microprocessors Make Intelligent Data Terminal Systems," *EE/Systems Engineering Today*, Vol. 33, No. 1 (January 1974), pp. 78-80. Part of "Microprocessors Course."

[25] Hoff, M.E., Jr., "Applications for Microcomputers in Instrumentation," *IEEE International Convention Technical Papers* (1973), Paper No. 21/1.

[26] Hoff, M.E., Jr., "Considerations for the Use of Microcomputers in Small Systems Design," *Wescon Technical Papers* (1972), Paper 26/3.

[27] Hoff, M.E., Jr., "New LSI Components," *6th IEEE Computer Society International Conference Digest* (1972), pp. 141-143.

[28] Hoff, M.E., Jr., "The One Chip CPU – Computer or Component? " *Proceedings, Computer Systems Design Conference* (1972), pp. 16-23.

[29] Holt, R.M., "Architecture and Organization of the AMI 7200 Microprocessor: A New Breed of Minicomputer," *IEEE Region 6 Conference on Minicomputers and Their Applications Conference Record* (1973), pp. 192-195.

[30] Horton, R.L., Englade, J. and McGee, G., "I^2L Takes Bipolar Integration a Significant Step Forward," *Electronics*, Vol. 48, No. 3 (February 6, 1975), pp. 83-90.

[31] Howick, J.F., "Study of Message Routing Within a Microcomputer Network," AD 756573 (December 1972), 45 p., Available from Technical Information Service, Springfield, Virginia 22151.

[32] Jackson, R.E., "Designing a Microprocessor-Based System," *EE/Systems Engineering Today*, Vol. 32, No. 12 (December 1973), pp. 62-67. Part of "Microprocessors Course."

[33] Kerr, H., "Microprocessors in Graphic Display Systems," *EE/Systems Engineering Today*, Vol. 32, No. 12 (December 1973), pp. 70-71. Part of "Microprocessors Course."

[34] Kuo, C., Kitagawa, N., Ogden, G., Hewkin, J., "16k RAM Built with Proven Process may Offer High Start-up Reliability."

[35] LEE ED, "When Programming Microprocessors, Use Your Hardware Background," *Electronics* (July 8, 1976), pp. 93-100.

[36] Lewis, D.R. and Siena, W.R., "How to Build a Microcomputer," *Electronic Design*, Vol. 21, No. 19 (September 13, 1973), pp. 60-65.

[37] Lewis, D.R. and Siena, W.R., "Microprocessor or Random Logic? " *Electronic Design*, Vol. 21, No. 18 (September 1, 1973), pp. 106-110.

[38] Lofthus, A., Ogden, D., "16 Bit Processor Performs Like a Minicomputer," *Electronics* (May 27, 1976), pp. 99-105.

[39] Metzger, J., "Forum: It's Go for Microprocessors," *Electronic Products*, Vol. 16, No. 6 (November 19, 1973), pp. 114-127.

[40] "Microcomputers Aim at a Huge New Markt," *Business Week*, No. 2279 (May 12, 1973), pp. 180-182.

[41] "Microprocessor-Driven CRT System Can Be User-Customized," *Computerworld* (September 13, 1972). Abstract in *SIGMICRO Newsletter*, Vol. 3, No. 3 (October 1972), p. 74.

[42] Monico, J.A. (Microsystems International, Ottawa, Canada), "Microprocessors – A Case History of an Integrated Approach," Presented at IEEE International Convention (March 26-29, 1974).

[43] Murphy, J.P. and others, "Enhancing an LSI Computer to Handle Decimal Data," *Electronics*, Vol. 46, No. 5 (March 1, 1973), pp. 77-82.

[44] Ogdin, J.L., "Survey of Microprocessors Reveals Limitless Variety," *EDN*, Vol. 19, No. 8 (April 20, 1974), pp. 38-43.

[45] Parasuraman, B., "Applications of LSI Processors," *Wescon Technical Papers* (1973), Paper No. 11/2.

[46] Pittman, T., "Improve Interrupt Handling Capability of Microprocessor with a Few ICs," *Electronic Design*, Vol. 24 (November 22, 1974).

[47] Reed, J.A., "Application of LSI Microprocessors to Display Terminals," *IEEE International Convention Technical Papers* (1973), Paper No. 21/2.

[48] Reyling, G., Jr., "Extend LSI Processor Capabilities," *Electronic Design*, Vol. 22 (October 25, 1974).

[49] Reyling, G., Jr., "LSI Building Blocks for Parallel Digital Processors," *IEEE International Convention Technical Papers* (1973), Paper No. 21/3.

[50] Reyling, G., Jr., "Microprocessors: The Next Generation in Digital Design," *EE/Systems Engineering Today*, Vol. 32, No. 11 (November 1973), pp. 86-90. Part of "Microprocessors Course."

[51] Reyling, G., Jr., "Performance and Control of Multiple Microprocessor Systems," *Computer Design*, Vol. 13, No. 3 (March 1974), pp. 81-86.

[52] Schmid, H., "Monolithic Processors," *Computer Design* (October 1974).

[53] Schultz, G.W., "Designing Optimized Microprogrammed Control Section for Microprocessors," *Computer Design*, Vol. 13, No. 4 (April 1974), pp. 119-124.

[54] Schultz, G.W. and others, "Guide to Using LSI Microprocessors," *Computer*, Vol. 6, No. 8 (June 1973), pp. 13-19.

[55] Seabury, T., "Microprocessors Signal in Traffic Control," *EE/Systems Engineering Today*, Vol. 33, No. 1 (January 1974), pp. 76-78. Part of "Microprocessors Course."

[56] Sideris, G., "Microcomputers Muscle In," *Electronics*, Vol. 46, No. 5 (March 1, 1973), pp. 63-64.

[57] Smith, H., "Anatomy of a Microcomputer," *EE/Systems Engineering Today*, Vol. 32, No. 11 (November 1973), pp. 91-94. Part of "Microprocessors Course."

[58] Smith, H., "Impact of Microcomputers on the Designer," *Wescon Technical Papers* (1973), Paper No. 11/1.

[59] Swales, N.P. and Weisbecker, J.A., "COSMAC – A Microprocessor for Minimum Cost Systems," *IEEE International Convention Technical Papers* (1974), Paper No. 17/2.

[60] Torrero, E.A., "Focus on Microprocessors," *Electric Design*, Vol. 18 (September 1, 1974).

[61] Theis, D.J., "Microprocessor and Microcomputer Survey," *Datamation* (December 1974).

[62] Weisbecker, J., "Simplified Microcomputer Architecture," *Computer*, Vol. 7, No. 3 (March 1974), pp. 41-47.

[63] Weiss, C.D., "Basic Microcomputer Software," *Electronic Design*, Vol. 22, No. 9 (April 26, 1974), pp. 142-146.

[64] Weiss, C.D., "Microcomputer Programming I: Basics," Notes for *NEC Professional Growth in Engineering Institute* (November 1973).

[65] Weiss, C.D., "Microcomputer Programming II; Producing Software," Notes for *NEC Professional Growth in Engineering Institure* (November 1973).

[66] Weiss, C.D., "MOS/LSI Microcomputer Coding," *Electronic Design*, Vol. 22, No. 8 (April 12, 1974), pp. 66-71.

[67] Weiss, C.D., "Software for MOS/LSI Microprocessors," *Electronic Design*, Vol. 22, No. 7 (April 1, 1974), pp. 50-57.

[68] Weissberger, A.J., "Distributed Function Microprocessor Architecture," *Computer Design* (November 1974).

[69] Weissberger, A.J., "MOS/LSI Microprocessor Selection," *Electronic Design*, Vol. 12 (June 7, 1974).

[70] Weissberger, A., Irwin, J., Kim, S.N., "Processorfamiliy Specializes in Dedicated Control," *Electronics* (July 8, 1976), pp. 84-89.

[71] Wickes, E., "8 Bit Microprocessor Aims at Control Application," *Electronics* (June 10, 1976), pp. 101-105.

[72] Wolf, H., "Microcomputers – A New Revolution," *EE/Systems Engineering Today*, Vol. 33, No. 1 (January 1974), pp. 80-83. Part of "Microprocessors Course."

WICHTIGE BEGRIFFE

Abtast- und Halteschaltung *(sample-and-hold)*
Eine Schaltung, die während eines kurzen Zeitintervalls eine Spannung abtastet und den Spannungspegel sehr genau für eine sehr viel längere Zeit speichert.

ACK *(acknowledge message)*
Quittungsmeldung, die über eine Nachrichtenverbindung gesandt wird, um den Empfang von korrekten Daten anzuzeigen. Sie wird in Verbindung mit Fehlerdetektoren in sog. Block- und Baumcodes verwendet.

Addierwerk *(adder)*
Digitalschaltung, die binäre Information in der Weise verknüpft, daß Summe und Übertrag gebildet werden.

Adresse *(address)*
Ein normalerweise numerischer Ausdruck, der einen spezifischen Speicherplatz kennzeichnet.

Adresse, indizierte *(indexed address)*
Eine Adresse, die mittels des Inhaltes eines Indexregisters vor oder während der Ausführung eines Befehles modifiziert wird.

– **relative** *(relative address)*
Die Zahl, die die Differenz zwischen der absoluten Adresse und der Basisadresse angibt.

– **unmittelbare** *(immediate address)*
verknüpft mit einer Instruktion, in welcher die Adresse, die den Wert des Operanden unmittelbar beinhaltet, enthalten ist.

Adressenformat *(address format)*
1. Anordnung des Adreßteils eines Befehls. Der Ausdruck "+1" wird oft benutzt um anzuzeigen, daß eine der Adressen den Speicherplatz des nächsten auszuführenden Befehls angibt.
2. Anordnung der Teile einer einzelnen Adresse, wie sie z.B. für die Identifizierung von Kanal, Modul, Spur usw. in einem Plattensystem benötigt wird.

Adressierung, direkte *(direct addressing)*
Eine Adressierungsart, bei der die Adresse des Operanden im Befehl oder nachfolgenden Speicherplatz enthalten ist.

– **indirekte** *(indirect addressing)*
Eine Adressierungsart, bei der die vorhandene Adresse den Speicherplatz eines Wortes angibt, das eine andere Adresse enthält. Diese indirekte Adresse wird dann dazu verwendet, den Operanden zu ermitteln.

Adreßregister *(address register)*
Ein Register, in dem Adressen gespeichert werden.

Akkumulator *(accumulator)*
Ein Register, welches einer arithmetisch/logischen Einheit zugeordnet ist und zur Speicherung von Operanden und Ergebnissen dient.

Algorithmus *(algorithm)*
Ein von den Mathematikern benutzter Begriff für einen Satz von Prozeduren, durch den man ein bestimmtes Ergebnis gewinnt.

Allzweck-Rechner *(general purpose computer)*
Ein Rechner, der für die Lösung einer großen Vielzahl von Problemen geeignet ist.

Alphanumerischer Code *(alphanumeric code)*
Ein Code, der aus Buchstaben, Ziffern und zugehörigen Sonderzeichen besteht.

ALU *(arithmetic logic unit)*
Eine arithmetische/logische Einheit oder auch ALU, ist Teil eines Rechenwerkes, in welchem arithmetische und logische Verknüpfungen von Operanden vorgenommen werden.

Analogdarstellung *(analog representation)*
Eine Darstellung nicht mittels diskreter Werte, sondern durch eine kontinuierliche Variable.

Arbeitsregistersatz *(working register set)*
Ein meist innerhalb des Rechenwerks untergebrachter Satz aus mehreren schnellen Registern zur Zwischenspeicherung von Operanden, Ergebnissen und Adressen.

Arbeitsweise, parallele *(parallel operation)*
Die Organisation der Datenmanipulierung in Bausteinen, bei der alle Ziffern eines Wortes gleichzeitig auf getrennten Leitungen übertragen werden, um die Arbeitsgeschwindigkeit zu erhöhen.

– **serielle** *(serial operation)*
Die Organisation der Datenmanipulation innerhalb von Bausteinen, bei der die Ziffern eines digitalen Wortes nacheinander auf einer einzelnen Leitung übertragen werden. Die serielle Arbeitsweise ist langsamer als die parallele, aber sie benötigt weniger komplexe Schaltungen.

Architektur-Mikroprozessor *(microprocessor architecture)*
Die Mikroprozessorarchitektur beschreibt das Zusammenwirken der Baugruppen eines Mikroprozessors mit dem Rechen- und Steuerwerk, mit den dazugehörigen Datenwegen und Steuerleitungen.

Architektur-Mikrocomputer *(microcomputer architecture)*
Die Mikrocomputerarchitektur beschreibt das Zusammenwirken der Baugruppen eines Mikrocomputers mit dem Mikroprozessor, dem Speicher (Daten/Programm) und der Ein-/Ausgabeeinheit mit den dazugehörigen Datenwegen und Steuerleitungen.

Architektur-Namen *(architecture name)*
Der Architekturname gibt an, wie die Operanden mittels eines Befehls eines Rechners (Mikrocomputers) miteinander verknüpft werden können und wo das Ergebnis abgelegt werden kann. Es wird zwischen drei Architekturen unterschieden:

a) Register zu Register-Architektur: mittels eines Befehls können nur Operanden in Registern verknüpft werden. Das Ergebnis wird wiederum in einem Register abgelegt.
b) Register zu Speicher-Architektur: mittels eines Befehls kann ein Operand aus dem Speicher mit einem Operand aus einem Register verknüpft werden. Das Ergebnis wird wiederum in einem Register abgelegt.
c) Speicher zu Speicher-Architektur: mittels eines Befehls können zwei Operanden aus dem Speicher miteinander verknüpft werden und das Ergebnis wird wiederum im Speicher abgelegt.

Die Speicher zu Speicher-Architektur schließt jeweils a) und b) mit ein. Die Register-Speicher-Architektur schließt a) mit ein.

ASCII *(American National Standard Code for Information Interchange)*
Ein genormter Code, der aus 7-Bit-Zeichen besteht (ev. mit einem 8. Bit für Paritätsprüfung). Er wird für den Informationsaustausch zwischen Datenverarbeitungssystemen, Nachrichtensystemen und den zugehörigen Geräten verwendet. Der ASCII-Code besteht aus Steuerzeichen und grafischen Zeichen. Er wird auch als USASCII bezeichnet.

Assembler *(assembler)*
Ein Programm, das assemblieren kann.

Assembler, resident *(resident assembler)*
Ein Programm zur Assemblierung, welches auf einem identischen Mikrocomputer lauffähig ist, für den auch die Assemblierung vorgenommen wird.

Assembler, cross *(cross assembler)*
Ein Programm zur Assemblierung, welches auf einer Großrechenanlage lauffähig ist. Da es meist in FORTRAN geschrieben ist, ist es meist anlangenunabhängig.

Asynchrone Einheit *(asynchronous device)*
Eine Einheit, deren Arbeitsgeschwindigkeit keinen Bezug zu irgendeiner Frequenz eines Systems hat, an das die Einheit angeschlossen ist.

Assemblieren *(assemble)*
Übersetzen eines Programms, das in einer symbolischen Sprache geschrieben ist (Assemblerprogramm), in die Maschinensprache. Dabei werden der symbolische Befehlscode durch einen absoluten Befehlscode und die symbolischen Adressen durch absolute oder verschiebbare (relocatable) Adressen ersetzt.

Baud *(baud)*
Eine Einheit für die Geschwindigkeit bei der Datenübertragung. 1 Baud = 1 Bit/Sekunde.

Baudot-Code *(baudot-code)*
Ein in der Datenübertragung verwendeter Code.

BCD-Code *(binary coded decimal)*
Ein binärer Code für die Darstellung von Dezimalzahlen in Gruppen von vier Bit. Der binäre Wert dieser 4-Bit-Gruppen reicht von 0000 bis 1001 und codiert die Zahlen "0" bis "9". Um bis 9 zu zählen, benötigt man 4 Bit, um bis 99 zu zählen, benötigt man zwei Gruppen von 4 Bit, bis 999 braucht man drei Gruppen von je 4 Bit usw.

Befehl *(instruction)*
Eine Anweisung, die eine Operation und die Werte oder Speicherplätze des dazugehörigen Operanden spezifiziert.

Befehlsphase ausführen *(instruction execute)*
Der Abschnitt eines Maschinenzyklus, in dem ein Befehl ausgeführt wird.

Befehlsphase holen *(instruction fetch)*
Der Abschnitt des Maschinenzyklus, in dem der nächste Befehl vom Speicher geholt wird.

Befehlsregister *(instruction register)*
Ein Register, das den Befehl zum Zwecke der Ausführung speichert.

Benchmark-Programm *(benchmark program)*
Ein Programm, um die Leistungsfähigkeit der Hard- und Software zu beurteilen und damit verschiedene Mikroprozessoren zu vergleichen.

Betriebssystem *(operating system)*
Die Software, die die Ausführung von Computerprogrammen steuert und die die Programmverwaltung, die Fehlersuche, die Ein/Ausgabeverwaltung, die Buchhaltung, das Übersetzen, die Speichereinteilung, das Datenmanagement und ähnliche Dienste übernehmen kann.

Block *(block)*
1. Ein Satz von Dingen wie Worte, Zeichen, Ziffern, die als eine Einheit angesehen werden.
2. Eine Ansammlung von benachbarten Datensätzen, die als eine Einheit abgespeichert ist. Blocks werden durch eine Blocklücke getrennt und jeder Block kann einen oder mehrere Datensätze enthalten.
3. Eine Gruppe von Bits oder Ziffern aus jedem beliebigen Zahlensystem, die als eine Einheit übertragen werden. Zu Zwecken der Fehlerkontrolle wird im allgemeinen die Gruppe von Bits oder Ziffern verschlüsselt.
4. Eine Gruppe von benachbarten Zeichen, die als eine Einheit gespeichert ist.

Blockdiagramm *(block diagram)*
Das Diagramm eines Systems, Gerätes oder Rechners, in welchem die wichtigsten Funktionsteile durch passend angeordnete geometrische Figuren dargestellt werden, um sowohl die wesentlichen Funktionen als auch das Zusammenspiel der einzelnen Teile aufzuzeigen.

Bootstrap-Lader *(bootstrap loader)*
Ein Programm bzw. ein bestimmtes Gerät, welches sich selbständig in einen gewünschten Zustand durch eigene Aktionen bringt, z.B. eine Routine, bei der die ersten Befehle bewirken, den Rest der Programme selbst in den Rechner von einem Eingabegerät zu laden.

Bus *(bus)*
Eine oder mehrere Leitungen zum Übertragen von Signalen oder Energie.

Byte *(byte)*
Eine Folge von aneinandergereihten binären Ziffern, die als eine Einheit angesehen werden. Gewöhnlich kürzer als ein Rechnerwort (in der Regel 8 Bit)

Carry *(carry)*
1. Ein oder mehrere Bits, die in Abhängigkeit von der Art arithmetischer Operationen gesetzt werden.
2. Die Zahl, die durch die oberen Bits (1.) repräsentiert wird.
3. Meistens ein Bit, definiert nach Punkt 1, das gesetzt wird, wenn die Summe bzw. das Produkt von zwei oder mehreren Bits gleich der höchsten Zahl des vorgesehenen Zahlenbereichs wird bzw. diesen überschreitet (Übertrag; Carry out = Überlauf, carry in = Übertrag).
4. Weniger allgemein, ein Borrow (Unterlauf).

Carry, vorausschauender *(carry look ahead)*
Ein Paralleladdierer, bei dem die Carry-Ausgänge jedes Volladdierers (jeder Stufe) vorausschauend durch die Prüfung der Operandeneingänge produziert werden.

Code *(code)*
1. Ein Satz von eindeutigen Vorschriften, in welcher Art und Weise Daten dargestellt werden können.
2. In Fernkommunikationssystemen ein System von Gesetzen und Konventionen, das festlegt, wie die Signale, die Daten repräsentieren, erzeugt, übertragen, empfangen und verarbeitet werden müssen.

3. In der Datenverarbeitung die Darstellung von Daten bzw. Rechner-Programmen in einer symbolischen Form, welche der Prozessor akzeptiert.

Compiler *(compiler)*

Ein Programm, das ein in einer höheren Programmiersprache geschriebenes Programm in die Maschinen- oder Assemblersprache übersetzen kann.

Computer *(computer)*

Ein elektronisches System, bestehend aus einer Zentraleinheit (CPU), Programm- und Datenspeicher und Ein/Ausgabeeinheiten, welches in der Lage ist digitale Informationen zu speichern, zu verarbeiten und ein- bzw. auszugeben.

CPU *(central processor unit)*

Zentraleinheit. Teil eines Computersystems, das ein Rechenwerk mit speziellen Registern und ein Steuerwerk enthält. Sie führt arithmetische Operationen durch, steuert den Befehlsablauf, stellt Taktsignale zur Verfügung und erledigt andere Verwaltungsfunktionen.

Daten *(data)*

1. Eine Darstellung von Fakten, Konzepten oder Befehlen in einer formalisierten Weise, die sich für die Kommunikation, Interpretation oder Verarbeitung durch menschliche oder maschinelle Mittel eignet.
2. Jede Darstellungsweise wie z.B. Zeichen oder analoge Größen, denen eine Bedeutung zugemessen wird oder zugemessen werden könnte.

Datenbus *(data bus)*

Ein für die Ein- und Ausgabe benutzter Weg eines Systems, auf dem Daten in das oder aus dem digitalen System gebracht werden können, wobei der gemeinsame Bus mit mehreren Subsystemen verbunden ist.

Datenverarbeitung *(data processing)*

Die Ausführung einer systematischen Folge von Operationen, deren Gegenstand Daten sind. Synonym mit Informationsverarbeitung.

Debug *(debug)*

Das Entdecken, Lokalisieren und Entfernen von Fehlern in einer Routine oder von Defekten in einem Computer. Debug-Programm = Fehler-Prüf- und -Erkennungsprogramm.

Decoder *(decoder)*

Ein Umsetzer-Baustein, der digitale Informationen empfängt – im Falle der Speicheransteuerung eine binäre Adreßinformation – die auf einer kleinen Anzahl von Leitungen erscheint und der daraufhin eine aus einer großen Zahl von Ausgangsleitungen auswählt und aktiviert.

Dezimal *(decimal)*

1. Bezieht sich auf ein Kennzeichen oder eine Eigenschaft, die eine Auswahl oder Bedingung beinhaltet, bei denen es zehn Möglichkeiten gibt.
2. Bezieht sich auf ein Zahlensystem auf der Basis der Zahl 10.

Dezimalziffer *(decimal digit)*

In dezimaler Darstellung eines der Zeichen 0 bis 9.

Digitalisieren *(digitize)*

Das Verwenden von numerischen Zeichen, um Daten auszudrücken oder darzustellen, d.h. das Gewinnen einer digitalen Darstellung einer Größe aus einer

analogen Darstellung einer physikalischen Größe.

Dump *(dump)*

Eine Kopie oder ein Auszug eines Speicherinhaltes.

Duplex *(duplex)*

Eine, die Arbeitsweise eines Bausteins in der Nachrichtenübertragung kennzeichnende Eigenschaft, bei der jede Endstation gleichzeitig senden und empfangen kann.

Durchgangs-Liste *(pushup list)*

"Als erster hinein – als erster heraus". Der jeweils älteste eingegebene Listenposten wird bei Verarbeitung als erster gelesen.

EAROM *(electrically alterable ROM)*

Ein Halbleiter Nur-Lesespeicher (ROM), welcher elektrisch löschbar und wieder programmierbar ist (elektrisch veränderbar).

EBCDIC *(extended binary codes decimal interchange code)*

Ein 8-Bit-Code mit 256 Zeichen, der für die Übertragung von binären Daten verwendet wird.

Ebene *(level)*

Niveau der Unterordnung in einer Hierarchie.

Echtzeit *(real time)*

1. Bezieht sich auf die aktuelle Zeit während der ein physikalischer Prozeß abläuft.
2. Bezieht sich auf die Durchführung einer Rechnung in einem Computer während der aktuellen Zeit, in der ein physikalischer Prozeß abläuft, um die Ergebnisse der Rechnung zur Führung des Prozesses zu verwenden.

ECL-Schaltungen *(emitter-coupled logic)*

Bipolare Emitter-gekoppelte Logik bzw. bipolare Schaltkreistechnologie.

Editieren *(edit)*

Form oder Format von Daten modifizieren, z.B. Einfügen oder Streichen von Zeichen.

Editor *(editor)*

Ein Computerprogramm, das es erlaubt über ein Eingabegerät Zeichen innerhalb eines Datenblocks (file) zu modifizieren, einzufügen oder zu entfernen.

Ein/Ausgabegeräte *(input/output device)*

Teile eines Computersystems, mit dem Daten in ein digitales System ein- oder ausgegeben oder durch die Daten für sofortigen oder zukünftigen Gebrauch festgehalten werden können.

Einsprungpunkt *(entry point)*

Ein gekennzeichneter Platz in einer Routine, der von außen angesprungen werden kann.

Emulieren *(emulate)*

Das funktionelle Nachbilden eines Systems durch ein anderes, so daß das nachbildende System die gleichen Daten erhält, die gleichen Programme ausführt und die gleichen Ergebnisse erzielt wie das nachgebildete System.

Emulator-In circuit *(in circuit emulator)*

Ein System, welches ein Mikroprozessor funktionell, elektrisch und mechanisch nachbilden kann. Die Verbindung wird mittels Steckers und eines Sockels erstellt.

EPROM *(erasable PROM)*

Ein Nur-Lesespeicher, welcher elektrisch programmierbar und durch UV-Licht wieder löschbar ist.

Exklusiv-ODER *(exclusive OR)*

Eine modifizierte Form der ODER-Funktion, die eine logische Gleichung entsprechend dem Summenausgang des Halbaddierers hat.

Exponent *(exponent)*

Ein bei der Gleitkommadarstellung vorkommendes Ziffernpaar, das die Potenz angibt, zu der die Basis des Zahlensystems erhoben wird.

Fan-In *(fan in)*

Gibt die Anzahl der normierten Eingangslasten eines Einganges an mit dem ein treibender Ausgang belastet wird. Beispiel: Fan-In = 1 stellt bei Standard-TTL eine Belastung eines Ausgangs von −1,6 mA im Low-Zustand ($V_{IL} \leqslant 0{,}8V$) und 40 μA im High-Zustand ($V_{IH} \geqslant$ 2,4 V) dar.

Fan-Out *(fan out)*

Gibt die Anzahl der normierten Lasten an, die an den Ausgang eines logischen Bausteins angeschlossen werden können (eine Last besteht normalerweise aus der Eingangsimpedanz eines logischen Bausteins; Ausgangsbelastbarkeit). Beispiel: Fan-Out = 10 bei Standard-TTL bedeutet, daß 10 normierte Eingangslasten (siehe Fan-In) angeschlossen werden können.

Fehler *(error)*

Irgendeine Diskrepanz zwischen errechneten, beobachteten oder gemessenen Werten und dem wahren, festgelegten, theoretisch korrekten Wert bzw. Bedingung.

Fehlererkennung/Korrektur *(ECD = error correction/detection)*

Erkennung und/oder Korrektur von Fehlern innerhalb eines Speicherwortes oder eines übertragenen Datenwortes/Block.

Feld *(field)*

Spezifizierter Bereich für spezielle Daten; z.B. eine Gruppe von Kartenspalten oder eine Anzahl von Bitstellen in einem Rechnerwort, die verwendet wird, eine Operandenadresse (adress field) oder Operationscode (operation field) oder Datenbereich (data field) auszudrücken.

Festkommazahl, binäre *(fixed-point binary number)*

Eine binäre Zahl, die durch ein Vorzeichenbit und ein oder mehrere Ziffernbits dargestellt wird, wobei das Komma irgendwo zwischen zwei benachbarten Bits fest definiert ist.

Festwertspeicher *(ROM = read only memory)*

Ein Halbleiterspeicher, der in der Fabrik mittels einer kundenspezifischen Metallisierungsmaske mit einem festen, nicht mehr änderbaren Programm programmiert worden ist.

Festwertspeicher, programmierbarer *(PROM = programmable read only memory)*

Ein Halbleiterspeicher mit festem Programm, der nur gelesen werden kann und der vom Anwender programmiert werden kann.

FIFO-Speicher *(FIFO)*

Siehe Silospeicher.

Flag *(flag)*

1. Irgendein Typ von Indikator für die Identifikation, z.B. Wertmarke;
2. Ein Zeichen, das das Eintreten einer Bedingung signalisiert, wie z.B. das Ende eines Datenwortes oder der den Überlauf (carry) bei einer Rechenoperation.

Flankentriggerung *(edge triggering)*

Aktivierung einer Schaltung, wobei die Flanke des Steuerpulses die Zustandsänderung der Schaltung bewirkt.

Flipflop *(flip flop)*

Eine Schaltung mit zwei stabilen Zuständen und der Fähigkeit, von einem Zustand in den anderen durch Anlegen eines Steuersignals zu wechseln, wobei nach Wegnehmen der Signale der Zustand erhalten bleibt.

Flußdiagramm *(flow chart)*

Eine grafische Darstellung für die Definition, Analyse oder Lösung eines Problems, bei der Symbole für die Wiedergabe der Befehle, der Daten, des Datenflusses, der Geräte usw. verwendet werden.

Format *(format)*

Die Anordung von Daten, Adressen oder Befehlen.

FORTRAN *(FORmula TRANslating system)*

Eine primär für die Darstellung von Rechnerprogrammen mittels arithmetischer Formeln verwendete höhere Programmiersprache.

Genauigkeit *(accuracy)*

Der Grad der Fehlerfreiheit, d.h. der Grad der Übereinstimmung mit der Wahrheit (Richtigkeit) oder mit einem Gesetz.

Genauigkeit, doppelte *(double precision)*

Bezieht sich auf die Verwendung von zwei Rechnerworten für die Darstellung einer Zahl bzw. von Ergebnissen, z.B. bei der Multiplikation von zwei 8-Bit-Zahlen ist das Ergebnis 16 Bit.

Gleitkommazahl, binäre *(floating-point binary number)*

Eine binäre Zahl, die in exponentieller Schreibweise dargestellt wird. D.h. ein Teil des binären Wortes repräsentiert die Mantisse und ein anderer Teil den Exponenten.

Halbaddierer *(half-adder)*

Eine logische Schaltung, die zwei binäre Zahlen addieren kann, aber ohne Berücksichtigung eines Übertrages aus einer vorangegangenen Addition (ohne carry-in).

Hamming-Code *(hamming code)*

Ein selbstkorrigierendes Codesystem, das bei der Datenübertragung verwendet wird. Dieser Code verwendet eine Paritätsprüf-Matrix.

Hardware *(hardware)*

Physikalisches Gerät im Gegensatz zu einem Rechnerprogramm (Software) oder einer Methode, z.B. mechanische, magnetische, elektrische oder elektronische Einrichtungen.

Hexadezimal-Code *(hexadecimal code)*

Ein binärer Code für die Darstellung von Zahlen in Gruppen zu 4 Bit mit der Basis 16. Der binäre Wert dieser Zahlen reicht von 0000 bis 1111, was den dezimalen Zahlen 0 bis 9 und den Buchstaben A, B, C, D, E und F entspricht (10 bis 15).

Indexregister *(index register)*

Ein Register, dessen Inhalt vor oder während der Ausführung eines Befehls zum (oder vom) Adreßoperand addiert (oder subtrahiert) werden kann.

Indizieren *(indexing)*

Eine in Rechnern verwendete Methode der Adreßmodifizierung mittels eines Indexregisters (siehe Adresse indizierte)

Informationssystem mit Rückmeldung *(information feedback system)*

Fernkommunikationssysteme mit "Echozeichen", die die Richtigkeit der Übertragung prüfen.

Interface *(interface)*

Eine gemeinsame Schnittstelle. Ein Interface kann eine Hardwarekomponente zur Verbindung zweier Geräte sein, oder es kann ein Bereich aus Speichereinheiten oder Registern sein, auf die von zwei oder mehreren Rechnerprogrammen zugegriffen wird (Software-Interface).

Interrupt *(interrupt)*

Unterbrechung eines Programms in der Art, daß es zu einem späteren Zeitpunkt wieder ohne Verlust von Informationen fortgesetzt werden kann.

Kanal *(channel)*

1. Eine Verbindung, über die Daten gesandt werden können, z.B. Datenkanal, Ausgabekanal.
2. Teil eines Speichermediums, auf das von einer gegebenen Schreib- oder Lesestation zugegriffen werden kann, z.B. Lochspur, Bandkanal.
3. Ein Hilfsmittel zur Übertragung in einer Richtung in der Nachrichtentechnik. Mehrere Kanäle können die gleichen gerätetechnischen Einrichtungen benutzen. Beispielsweise benutzt jeder Kanal in einem Trägersystem auf Frequenzmultiplex-Basis ein bestimmtes für ihn reserviertes Frequenzband.

Kaskadenschaltung *(cascade connection)*

Zwei oder mehrere gleichartige in Reihe geschaltete Bausteine, wobei der Ausgang des einen Bausteins mit dem Eingang des nächsten verbunden ist.

Komplement-Darstellung *(complement notation)*

Ein Darstellungssystem, in dem positive binäre Zahlen mit positiven Zahlen in Größe und Vorzeichen identisch sind, in dem aber die negativen Zahlen das genaue Komplement der Größe des entsprechenden positiven Wertes sind.

Lesen, zerstörungsfrei *(nondestructive read out)*

Liegt bei Speichern vor, die so konstruiert sind, daß ein Lesevorgang den gespeicherten Inhalt nicht ändert. Es ist nicht nötig, nach jedem Lesevorgang einen Schreibvorgang folgen zu lassen.

Lesezeit *(read time)*

Siehe Zugriffszeit.

Lichtgriffel *(light pen)*

Eine kleine Fotozelle oder ein Fotovervielfacher in einem griffelförmigen Gehäuse, der gegen einen Bildschirm gehalten wird, um den Zeitpunkt festzustellen, in dem der abgelenkte Elektronenstrahl durch den speziellen Ort läuft.

Logik, negative *(negative logic)*

Eine Logik, bei der die negativere Spannung den logischen Zustand "1" repräsentiert und entsprechend die positivere Spannung den Zustand "0" repräsentiert.

Logik, positive *(positive logic)*

Eine Logik, bei der die positivere Spannung den logischen Zustand "1" repräsentiert und entsprechend die negativere Spannung den Zustand "0" repräsentiert.

Löschen *(erase)*

Das Tilgen der in einem Speicher enthaltenen Information; überschreiben (siehe EAROM oder EPROM).

LSI *(large scale integration)*

Die Realisierung von Chips mit großer Fläche in Verbindung mit höchster Packungsdichte der Komponenten. Dies hat eine Kostenreduktion bei Maximierung der Anzahl von Systemverbindungen auf der Ebene des Halbleiterplättchens zur Folge. Die Komplexität beträgt mehr als 100 equivalente Gatterfunktionen (siehe SSI und MSI).

Makrobefehl *(macro instruction)*

Ein Befehl in einer Quellsprache, der äquivalent einer bestimmten Folge von Maschinenbefehlen ist.

Makroprogrammieren *(macro programming)*

Programmieren mit Makrobefehlen.

Mikrocomputer *(micro computer)*

Kleines integriertes System, welches die Funktion eines Computers ausführt und aus Zentraleinheit, Programm/Datenspeicher und Ein-/Ausgabe besteht.

Mikrobefehl *(micro instruction)*

Codierte Darstellung der zur Steuerung einer Zentraleinheit notwendigen Steuerinformationen (siehe Softwareebenen).

Mikroprogrammierung *(microprogramming)*

Eine Methode, die angewendet wird, um den Ablauf eines gespeicherten Maschinenbefehls zu steuern. Im allgemeinen wird bei dieser Technik ein programmierter Festwertspeicher (Mikroprogrammspeicher) verwendet, der mehrere oft gemeinsam auftretende Steuerfolgen enthält (siehe Softwareebenen).

Mikroprozessor *(microprocessor)*

Der integrierte Teil eines Computers ohne Programm-/Datenspeicher und Ein-/Ausgabe, d.h. die Zentraleinheit oder CPU, welche aus dem Rechen- und Steuerwerk besteht.

Mikroprozessorelement *(microprocessor bit slice)*

N-Bit Teil eines mikroprogrammierbaren Mikroprozessors, welches zu M · N-Bit kaskadierbar ist und keine Mikroprogrammablaufsteuerung enthält.

Marke *(label)*

Ein oder mehrere Zeichen zur Identifikation einer Anweisung oder eines bestimmten Datenwortes in einem Computerprogramm.

Maschinenbefehl *(machine instruction)*

Der Befehl, den ein Mikro)Computer erkennen und ausführen kann. (Folge von binären "0" und "1").

Maschinensprache *(machine language)*

Eine Sprache, die direkt von einem (Mikro)Computer verwendet wird.

Maske *(mask)*

1. Ein Muster von Zeichen, welches die Beibehaltung oder Elimination von Teilen eines anderen Zeichenmusters steuert.
2. Ein Filter.

Matrix *(matrix)*

1. Mathematisch ein 2dimensionales, rechteckiges Feld von Zahlen (Matrixalgebra).
2. In Rechnern, Logiknetzwerke in obiger Form, deren Knotenpunkte teilweise verbunden sind ("Diodenmatrix").
3. Ein Zahlenfeld beliebiger Dimension.

MODEM *(MOdulator - DEModulator)*

Ein Gerät, das über nachrichtentechnische Einrichtungen übertragene Signale moduliert und demoduliert.

MOS-Transistor *(MOS ≙ metal-oxide semiconductor)*

Ein aktives Halbleiterelement bei dem ein leitender Kanal in einer Region zwischen zwei Elektroden induziert wird, indem man eine Spannung an eine auf der Oberfläche dieser Region befindliche isolierte Elektrode anlegt.

MSI *(medium scale integration)*

Halbleiterbausteine mit mittlerem Integrationsgrad, z.B. Zähler. Entsprechend 40 bis 100 Gatterfunktionen.

Multiplexen *(multiplex)*

Das gleichzeitige oder überlappende Übertragen mehrerer Signale auf einem einzigen Kanal.

Nachrichtenverbindung *(communication link)*

Die physikalischen Einrichtungen, die einen Ort mit einem anderen verbinden, um das Übertragen oder Empfangen von Daten zu ermöglichen.

NAK-Signal *(not acknowledge)*

Meldung, die über Datenkommunikationsverbindungen gesendet wird, wenn die empfangenen Daten fehlerhaft sind und ein zweites Mal übertragen werden sollen.

Netzwerk, kombinatorisches *(combinatorial logic system)*

Ein digitales System, das keine Speicherelemente enthält.

Notizblockspeicher *(scratch-pad memory)*

Ein kleiner Speicher- (Register-)bereich, der für die lokale Datenverarbeitung auf Zwischenspeicherbasis benutzt wird.

Objektcode *(object code)*

Ergebnis eines Compilers oder Assemblers, das aus einem ausführbaren Maschinencode besteht, oder das für eine Weiterverarbeitung zum Erzeugen eines ausführbaren Maschinencodes geeignet ist.

Objektsprache *(object language)*

Die Sprache, in die eine Anweisung übersetzt wird.

OCR *(optical character recognition)*

Optische Zeichenerkennung.

Octal-Code *(octal code)*

Ein binärer Code für die Darstellung von Zahlen in Gruppen zu 3 Bit mit der Ba-

sis 8. Der binäre Wert dieser Zahlen reicht von 000 bis 111 (0 bis 7).

Operand *(operand)*

Objekt einer Anweisung oder eines Befehls. Ein Operand wird gewöhnlich durch den Adreßteil eines Befehls identifiziert.

Operation *(operation)*

1. Eine definierte Aktion bzw. ein Satz von Aktionen, um ein Ergebnis einer Verknüpfungsvorschrift von Operanden zu erhalten.
2. Die Vorschrift selbst.
3. Die Aktion, festgelegt durch eine einfache Computer-Instruktion.
4. Ein Programmschritt, ausgeführt durch einen Computer, z.B. Addition, Subtraktion usw. Die Operation wird gewöhnlich durch das Operationsfeld spezifiziert.
5. Die Aktion, die ein logisches Element ausführen kann.

Operationscode *(operation code)*

Ein Code, der eine bestimmte Operation repräsentiert. Synonym mit Befehlscode.

Optische Zeichenerkennung (OCR) *(optical character recognition)*

Die maschinelle Identifikation von gedruckten Zeichen mittels lichtempfindlicher Geräte.

Packen *(pack)*

Das Komprimieren von Daten in einem Speichermedium, indem man bekannte Eigenschaften der Daten ausnutzt, so daß die ursprünglichen Daten rekonstruiert werden können. Z.B. das Komprimieren von Daten in einem Speichermedium durch Verwendung von Bits oder Bytes, die andernfalls unbenutzt blieben.

Parameter *(parameter)*

Eine Variable, die für einen bestimmten Zweck oder einen bestimmten Prozeß einen konstanten Wert bekommt.

Paritätsbit *(parity bit)*

Ein Prüfbit, das an ein Feld von binären Ziffern angehängt ist, um die Quersumme aller binären Ziffern einschließlich des Prüfbits immer gerade oder immer ungerade zu machen.

Paritätsprüfung *(parity check)*

Die Technik, ein Bit zu einem digitalen Wort hinzuzufügen, um die Gesamtzahl von binären Einsen und Nullen entweder immer gerade oder immer ungerade zu machen. Diese Prüfmethode zeigt einen Fehler in den Daten an, aber nicht den Ort des Fehlers.

Peripheriegeräte *(peripheral equipment)*

Einheiten, die zusammen mit einem Computer arbeiten, aber kein Teil von ihm sind.

PLA *(programmable logic array)*

Eine integrierte Schaltung, die ROM-Matrizen enthält, um Summen- und Produkt-Terme eines logischen Netzwerkes zu kombinieren.

Programm *(program)*

1. Eine Reihe von Aktionen, die vorgesehen sind, um ein bestimmtes Ergebnis zu erzielen.
2. Grob gesagt: eine Routine.

Programmschrittzähler *(program counter)*

Ein Register oder Zähler, welches die Adresse des Befehls beinhaltet, der als nächstes ausgeführt wird. Nach erfolgter Ausführung wird es um 1 inkrementiert oder auf einen neuen Wert gesetzt (Sprung).

PROM *(PROM)*

Siehe Festwertspeicher.

Prozessor *(processor)*

1. In der Hardware: ein Datenprozessor.
2. In der Software: ein Rechnerprogramm, das einen Compiler, Assembler und ähnliche Funktionen für eine spezielle Programmiersprache beinhaltet, wie z.B. ein COBOL-Prozessor oder FORTRAN-Prozessor.

Prüfbit *(check bit)*

Ein binäres Prüfzeichen, z.B. ein Paritätsbit.

Puffer *(puffer)*

1. Ein rückwirkungsfreier Baustein, der benutzt wird, um einen Einfluß des angesteuerten Bausteins auf die ansteuernde Treiberschaltung zu vermeiden.
2. Eine Speichereinheit (Register), die zum Ausgleich von unterschiedlichen Datenflußgeschwindigkeiten oder unterschiedlichen Zeitpunkten des Auftauchens von Ereignissen bei einer Datenübertragung von einem Gerät zum anderen dient.

Punktmatrix *(dot matrix)*

Eine Matrix aus Punkten, die für die Darstellung von alphanumerischen Zeichen verwendet wird.

Quellenprogramm *(source program)*

Ein Computerprogramm, das in der Quellensprache geschrieben ist.

Quellensprache *(source language)*

Die Sprache, aus der eine Anweisung übersetzt wird.

RAM *(Random access memory)*

Siehe Schreib-/Lesespeicher.

Rauschen *(noise)*

Ein Begriff, der sich auf störende oder unerwünschte elektrische Signale bezieht.

Rechenwerk *(arithmetic logic unit)*

Teil einer Zentraleinheit eines Computers, welcher arithmetische und logische Operationen durchführt sowie allgemeine oder spezielle Register enthält, z.B. Akkumulator.

Redundanz *(redundancy)*

Die Technik, mehr als einen Baustein derselben Art für eine bestimmte Funktion einzusetzen oder innerhalb eines Codes die Verwendung von zusätzlichen Bits, die nicht zur reinen binären Darstellung notwendig sind.

Register *(register)*

Temporärer Speicher für digitale Daten.

ROM *(read only memory)*

Siehe Festwertspeicher.

Rückkopplungsschleife *(feedback loop)*

Die Bauteile und Prozesse, die beim Korrigieren oder Regeln eines Systems beteiligt sind, bei dem ein Teil des Ausgangssignal als Eingangssignal verwendet wird.

Schaltung, synchrone *(synchronous circuit)*

Eine Schaltung, bei der alle gewöhnlichen Operationen von gleichmäßig geformten Signalen eines Taktgenerators gesteuert werden.

Schalt- und Durchlaufverzögerung *(propagation delay)*

Die Zeit, die benötigt wird, um den Wechsel eines logischen Pegels durch ein Bauelement oder eine Kette von Bauelementen zu übertragen.

Schieben *(shift)*

Das Bewegen von Daten nach links oder nach rechts.

Schieben, arithmetisches *(shift arithmetic)*

1. Vorzeichenbit wird durch das Schieben nicht verändert.
2. Entspricht der Multiplikation um $2^{\pm n}$.

Schieben, logisches *(shift logic)*

Ein alle Stellen betreffender Schiebevorgang.

Schleife *(loop)*

Eine Folge von Befehlen, die wiederholt durchgeführt wird, bis eine Abbruchbedingung erfüllt ist.

Schieberegister *(shift register)*

Ein Register, in dem die Daten nach links oder nach rechts bewegt werden können.

Schreib-/Lesespeicher *(RAM $\hat{=}$ random-access memory)*

Ein Speicher, aus dem jede Information mit ungefähr der gleichen Zeitverzögerung gelesen werden kann, wobei die Adresse willkürlich gewählt werden kann, ohne daß man zunächst einen großen Wust irrelevanter Daten durchsuchen muß. Außerdem können in diesen Speicher beliebige Informationen eingeschrieben werden.

Schreibzeit *(write time)*

Die Zeit, während dem der richtige Pegel an den Adress-, Steuer- und Datenleitungen anliegen müssen, um das erfolgreiche Schreiben der Daten in den Speicher garantieren zu können.

Sequenz *(sequence)*

Eine sinnvolle Schrittfolge, z.B. bei einer Ablaufsteuerung.

Sequenzielles Logiksystem *(sequential logic system)*

Digitales System, welches Speicherelemente verwendet.

Silospeicher *(FIFO)*

Liest gespeicherte Daten nach der Methode first-in/first-out (zuerst herein/zuerst heraus) aus. Auch bekannt als FIFO-Speicher.

Simulator *(simulator)*

Ein Gerät, System oder Computerprogramm, das bestimmte Kennzeichen oder Verhalten eines physikalischen oder abstrakten Systems simuliert.

Simulieren *(simulation)*

1. Das Darstellen bestimmter Züge des Verhaltens eines physikalischen oder abstrakten Systems durch das Verhalten eines anderen Systems.
2. Das Darstellen von Gerätefunktionen, Systemen und Computerprogrammen durch andere Mittel, z.B. das Verhalten eines physikalischen Systems

durch den Ablauf eines Computerprogramms über ein mathematisches Modell darstellen.

Skippen *(skip)*

Das Ignorieren (überspringen) einer oder mehrerer Instruktionen in einer Folge von Instruktionen, meist als Folge einer erfüllten Bedingung. Beispiel: Skip if carry = 1; überspringe den nächsten Befehl, wenn carry = 1.

Software *(software)*

Ein Satz von Computerprogrammen, Prozeduren und der möglicherweise dazu gehörenden Dokumentation, der mit dem Arbeiten eines Datenverarbeitungssystems zusammenhängt, z.B. ein Compiler, Bibliotheksprogramme, Handbücher, Schaltungsdiagramme.

Speicher, flüchtiger *(volatile storage)*

Ein Speicher, dessen Daten bei Wegnahme der Spannungsversorgung verlorengehen, z.B. Halbleiterspeicher.

Speicher, nichtflüchtiger *(non-volatile storage)*

Ein Speicher, dessen Informationsinhalt auch nach Abschalten der Versorgungsspannung erhalten bleibt, z.B. Kernspeicher.

Speicher, temporärer *(temporary storage)*

Beim Programmieren reservierter Speicher für Zwischenergebnisse.

Speicherelemente, dynamische *(dynamic memory)*

Speicherelemente, deren Zellen innerhalb geeigneter Zeitintervalle aufgefrischt werden müssen, um einen Verlust des Informationsinhaltes zu vermeiden.

Speicherelemente, statische *(static memory)*

Speicherelemente mit Zellen, die ihre Information solange halten, wie die Spannungsversorgung anliegt, es sei denn, die Information wird durch äußere Anregung geändert.

Speicherzugriff, direkter *(DMA ≙ direct memory access)*

Ein Ein-/Ausgabeverfahren für ein System, das einen kleinen Prozessor besitzt, dessen einzige Aufgabe die Steuerung der Ein-/Ausgabe ist. Bei direktem Speicherzugriff werden Daten in das oder aus dem System gebracht, ohne daß das Programm eingreift.

Sprache *(language)*

Ein Satz von Abbildungen, Konventionen und Regeln zur Vermittlung von Information.

Sprache, problemorientierte *(problem oriented language)*

Eine Programmiersprache, die für die passende Darstellung einer bestimmten Klasse von Problemen geeignet ist, z.B. ALGOL für mathematische oder COBOL für kaufmännische Probleme.

Sprung *(jump, branch)*

Abweichung von der normalen Befehlsausführsequenz in einem Computer.

Sprung, bedingter *(conditional jump)*

Ein Sprung, der ausgeführt wird, falls bestimmte Kriterien erfüllt sind, z.B. Jump if Borrow; Springe, wenn Borrow gesetzt.

Sprungbedingungen *(jump conditions)*

Bedingungen, die in einer "Übergangs"-Tabelle festgelegt sind, welche die Än-

derung des Zustandes bestimmter Flipflops bestimmen.

SSI *(small scale integration)*

Halbleiterbausteine mit geringem Integrationsgrad, z.B. Gatterfunktionen.

Stack *(stack)*

Ein Registersatz oder Speicherbereich, welcher eine Stapel-Liste implementiert, d.h. die zuletzt eingeschriebene Information wird als erstes wieder ausgelesen.

Stack Pointer *(stack pointer)*

Register in dem die Adresse des Stacks gespeichert ist.

Stapel-Liste *(pushdown list)*

Eine Liste, die so angelegt und abgearbeitet wird, daß der Posten, der als erster wieder abgerufen wird, der ist, der als letzter in die Liste aufgenommen wurde (last in, first out).

Steuerung, numerische *(numerical control)*

Eine automatische Steuerung eines Prozesses, die mittels eines Gerätes durchgeführt wird, das von allen oder einem Teil von numerischen Daten Gebrauch macht, die während des Ablaufs des Prozesses eingegeben werden.

Steuergerät *(controller)*

Ein digitales Subsystem, das dafür verantwortlich ist, "wie" und "wann" ein System zu funktionieren hat.

Steuerhierarchie *(control hierarchy)*

Ein Entwurfskonzept, das in komplexen Systemen verwendet wird, um eine Prioritätsanordnung für verschiedene Steuereingriffe sicherzustellen, die von mehr als einer Quelle kommen.

Steuerzeichen *(control character)*

Ein Zeichen, dessen Auftauchen in einem bestimmten Textzusammenhang eine Steueroperation startet, modifiziert oder stoppt, wie z.B. ein Zeichen, das den Wagenrücklauf steuert oder ein Zeichen, das die Datenübertragung über ein Nachrichtennetz steuert. Ein Steuerzeichen kann gespeichert werden, um es in einer nachfolgenden Aktion zu verwenden. In manchen Fällen hat es eine graphische Darstellung.

Steuerwerk *(control)*

Teil einer Zentraleinheit eines Computers, welches den Ablauf von Befehlen und Datentransporten innerhalb eines Computers steuert.

Störabstand *(noise immunity)*

Ein Maß für die Unempfindlichkeit einer logischen Schaltung in bezug auf die Triggerung oder das Reagieren auf störende oder unerwünschte Signale oder Rauschen. Er wird stark durch den logischen Signalhub beeinflußt. Rauschen oder Übersprechen kann in positiver oder negativer Richtung auftreten.

Symbol *(symbol)*

Ein Symbol, das der Unterstützung des menschlichen Gedächtnisses dient, z.B. eine Abkürzung wie "mul" für "multipliziere".

System *(system)*

1. Eine Zusammenstellung von Methoden, Prozeduren oder Techniken, vereinigt durch festgelegte Wechselbeziehungen, die das Ganze zu einer organisierten Einheit formen.
2. Eine organisierte Sammlung von Menschen und Maschinen und Methoden, um ein Satz spezifischer Funktionen erfüllen zu können.

Terminal *(terminal)*

Ein Ort in einem System oder Kommunikationsnetzwerk, an dem Daten entweder ein- oder ausgegeben werden können. Synonym mit Datenstation.

TTL-Schaltung *(TTL ≙ transistor-transistor logic)*

Eine bipolare Halbleitertechnologie.

Übersetzen *(compile/translate)*

Das Erzeugen eines Programms in Maschinensprache aus einem Computerprogramm, das in einer anderen Programmsprache geschrieben ist. Dabei wird von der logischen Gesamtstruktur des Programms Gebrauch gemacht oder auch mehr als ein Maschinenbefehl für jeden symbolischen Befehl erzeugt. Ferner ist die Funktion des Assemblierens inbegriffen.

Übersprechen *(crosstalk)*

Eine in einem bestimmten Kanal auftretende Störung, die von einem anderen Kanal verursacht wird.

Unterprogramm *(subroutine)*

Ein Programm, das Teil eines anderen Programms sein kann.

USASCII *(United States of America Standard Code of Information Interchange)*

Ein in den Vereinigten Staaten verwendeter Standardcode für die Datenübertragung. Siehe ASCII-Code.

Variable *(variable)*

Eine Größe, die jeden Wert aus einem bestimmten Satz von Werten annehmen kann.

Verschlüsseln *(encode)*

Das Anwenden eines Satzes von eindeutigen Regeln, die spezifizieren, auf welche Weise Daten dargestellt werden können, so daß eine nachfolgende Entschlüsselung möglich ist. Synonym mit codieren.

Verzweigen *(branching)*

Eine Methode während des Programmablaufs, aufgrund von Resultaten die nächste auszuführende Operation anzuwählen. Siehe bedingter Sprung.

Volladdierer *(full adder)*

Ein logischer Schaltkreis ähnlich einem Halbaddierer, der jedoch einen Übertragseingang von einer vorangegangenen Addition verarbeiten kann.

Vorbereitungszeit *(setup time)*

Die minimal notwendige Zeit, die anliegende Daten vor dem Takten eines Bausteins an dessen Eingängen anliegen müssen, um die richtige Datenübernahme zu gewährleisten.

Wahrheitstafel *(truth table)*

Eine Tafel, auf der alle möglichen Zustände der Ein- und Ausgänge eines Schaltkreises tabuliert und zusammengefaßt sind. Sie zeigt an, was bei einer gegebenen Eingangskombination am Ausgang geschieht.

Wiederauffrischen *(refreshing)*

Das Auffüllen der durch nichtideale Isolation schwindenden Ladung auf einer Kapazität. Siehe Speicher, dynamische.

Wort *(word)*

Eine Zeichenreihe oder eine Bitreihe, die als eine Einheit angesehen werden, z.B. 4, 8 oder 16 Bit.

write enable

Auch read/write oder R/W genannt. Ein Steuersignal für ein Speicherelement oder einen Speicher, das den Schreibmodus aktiviert. Liegt umgekehrt der Schreibmodus nicht vor, dann ist der Lesemodus aktiviert.

Zähler *(counter)*

Eine Schaltung, die Eingangsimpulse zählt und nach Empfang einer vorbestimmten Zahl von Pulsen einen Puls am Ausgang abgibt.

Zeichen *(character)*

Ein Buchstabe, eine Zahl oder ein anderes Symbol, das als Teil der Organisation, Steuerung oder Darstellung von Daten verwendet wird. Ein Zeichen besteht oft aus einer räumlichen Anordnung von aufeinanderfolgenden oder verbundenen Strichen.

Zentraleinheit *(central processing unit)*

Siehe CPU.

Ziffer *(digit)*

Eine Ziffer, die eine der nichtnegativen ganzen Zahlen repräsentiert, die kleiner als die Basis des jeweiligen Zahlensystems ist. In dezimaler Darstellung ist z.B. eine Ziffer eines der Zeichen 0 bis 9. Synonym mit numerischem Zeichen.

Zugriff, direkter *(direct access)*

1. Verknüpft mit dem Prozeß, Daten zu holen bzw. abzuspeichern, wobei die Zeit für den Zugriff unabhängig von dem Ort, in dem sich das Datum befindet, ist.
2. Verknüpft mit dem Speichergerät, bei dem die Zugriffszeit effektiv unabhängig vom Ort des Datums ist.
3. Synonym für zufälligen Zugriff.

Zugriffszeit *(access time)*

Das Zeitintervall zwischen dem Absenden der Informationsanforderung und dem Erhalt dieser Information.

Zustand *(state)*

logische 1, logische 0. Technik in der Schaltungstechnik.

Zweierkomplement-Darstellung *(2's complement notation)*

Eine Schreibweise, in der positive binäre Zahlen identisch sind mit positiven Zahlen in Betrag und Vorzeichen, aber bei der eine 1 zum Einerkomplement addiert werden muß, um negative Zahlen zu erhalten.

Zyklus *(cycle)*

1. Ein Zeitintervall, in dem ein Satz von Ereignissen durchgeführt wird.
2. Ein Satz von Operationen, der in der gleichen Reihenfolge wiederholt wird. Die Operationen können eventuell bei jeder Wiederholung variiert werden.

Workshops von Texas Instruments

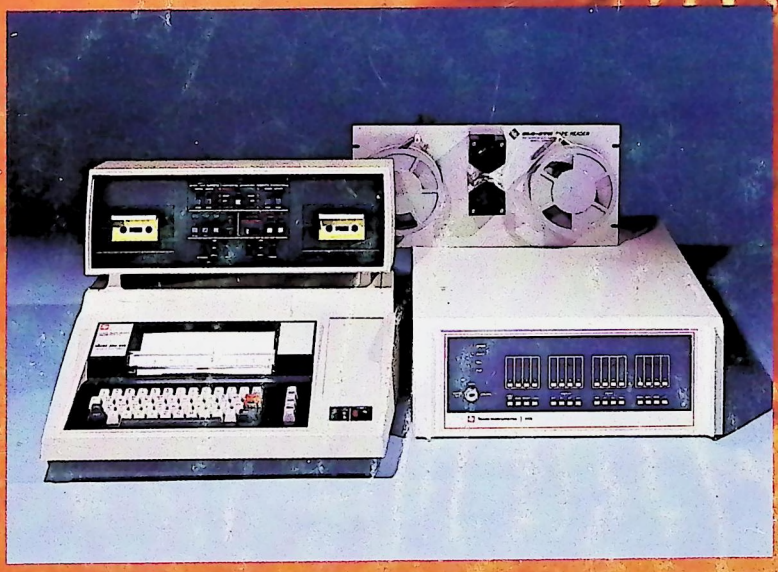
Workshops schließen die Know-how-Lücke zwischen Hersteller und Anwender. Jeder Anwender hat die Wahl, sich das umfangreiche und notwendige Fachwissen über Mikroprozessoren selbst zu erarbeiten oder es sich durch den Hersteller konzentriert vermitteln zu lassen. Die zweite Möglichkeit ist wirtschaftlicher. Dies gilt in den meisten Fällen auch dann, wenn beim Anwender bereits erfahrene Fachleute für Mikrocomputer zur Verfügung stehen. Denn, abgesehen vom dafür notwendigen Zeitaufwand, nicht jeder Fachmann ist auch in der Lage, seine Kenntnisse so weiterzugeben, daß sie schnell und nutzbringend angewendet werden können.

Workshops von Texas Instruments bieten ein hohes Maß an Objektivität, weil wir das breiteste Typenspektrum anbieten. Sie sind informativ und effektiv, weil Texas Instruments seit Jahren Klein- und Großcomputer, Taschenrechner, Mikroprozessoren und Mikrocomputer herstellt. Und weil die Teilnehmer (maximal 10 bis 15) durch erfahrene Mikroprozessor-Spezialisten mit Hilfe vieler Beispiele in praktischen Übungen geschult werden.

Der Grundlagen-Workshop (2 Tage) vermittelt den Teilnehmern in besonders übersichtlicher und absolut objektiver Form grundlegende Kenntnisse über Aufbau, Anwendung und Beurteilung von Mikroprozessoren und Mikrocomputern. In 3- und 5tägigen produktbezogenen Workshops werden die Teilnehmer soweit geschult, daß selbständiger Entwurf und Entwicklung von Systemen mit diesen Produkten inkl. Programmerstellung nach sehr kurzen Anlaufzeiten möglich werden.

Die ersten Workshops wurden im Sept. '76 durchgeführt. In diesem Jahr finden alle Workshops jeweils monatlich statt. Daß über 85% aller Teilnehmer die Workshops als gut oder ausgezeichnet beurteilen, empfinden wir als Ansporn, unsere Konzeption weiterzuverfolgen und unsere Kunden noch mehr als bisher zu unterstützen.

Das detaillierte Workshop-Programm erhalten Sie direkt von Texas Instruments Deutschland GmbH, Haggertystr. 1, 8050 Freising.



TEXAS INSTRUMENTS
Deutschland GmbH

ISBN 3-88078-009-9